Modelling and Analysis of Network Security Policies

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer Engineering (29^{th} cycle)

Modelling and Analysis of Network Security Policies

By

Fulvio Valenza

******

Supervisor(s):
Prof. Antonio Lioy, Supervisor
Ing. Cataldo Basile, Co-Supervisor

Doctoral Examination Committee:
Prof. Stefano Paraboschi, Referee, Università degli Studi di Bergamo
Prof. Herbert Leitold, Referee, A-SIT
Prof. Refik Molva, EURECOM
Prof. Riccardo Sisto, Politecnico di Torino
Prof. Guido Marchetto, Politecnico di Torino

Politecnico di Torino
2017

"Companies spend millions of dollars on firewalls, encryption, and secure access devices and it's money wasted because none of these measures address the weakest link in the security chain: the people who use, administer, operate and account for computer systems that contain protected information."

- Kevin David Mitnick, *March 2000*

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Fulvio Valenza
2017

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Antonio Lioy and my co-advisor Ing. Cataldo Basile, for their continuous support to my PhD study.

My special thanks also goes to my colleagues Daniele Canavese and Christian Pitscheider for their wonderful collaboration. They supported me greatly and were always willing to help me.

A sincere thanks is also for all the people who worked with me in the TORSEC research group of the Politecnico di Torino, in the CE&ENG group of the CNR. I would like also to thank the people who participated with me in the European project SECURED.

I would like to express my very profound gratitude also to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

The last but not the least words are powerless to express my gratitude to Serena for her love and encouragements!

Thank you very much, everyone!!!!

Fulvio

Abstract

Nowadays, computers and network communications have a pervasive presence in all our daily activities. Their correct configuration in terms of security is becoming more and more complex due to the growing number and variety of services present in a network.

Generally, the security configuration of a computer network is dictated by specifying the policies of the security controls (e.g. firewall, VPN gateway) in the network. This implies that the specification of the network security policies is a crucial step to avoid errors in network configuration (e.g., blocking legitimate traffic, permitting unwanted traffic or sending insecure data).

In the literature, an anomaly is an incorrect policy specification that an administrator may introduce in the network. In this thesis, we indicate as policy anomaly any conflict (e.g. two triggered policy rules enforcing contradictory actions), error (e.g. a policy cannot be enforced because it requires a cryptographic algorithm not supported by the security controls) or sub-optimization (e.g. redundant policies) that may arise in the policy specification phase.

Security administrators, thus, have to face the hard job of correctly specifying the policies, which requires a high level of competence. Several studies have confirmed, in fact, that many security breaches and breakdowns are attributable to administrators' responsibilities.

Several approaches have been proposed to analyze the presence of anomalies among policy rules, in order to enforce a correct security configuration. However, we have identified two limitations of such approaches. On one hand, current literature identifies only the anomalies among policies of a single security technology (i.e., IPsec, TLS), while a network is generally configured with many technologies. On the other hand, existing approaches work on a single policy type, also named domain (i.e., filtering, communication protection).

Unfortunately, the complexity of real systems is not self-contained and each network security control may affect the behavior of other controls in the same network.

The objective of this PhD work was to investigate novel approaches for modelling security policies and their anomalies, and formal techniques of anomaly analysis. We present in this dissertation our contributions to the current policy analysis state of the art and the achieved results.

A first contribution was the definition of a new class of policy anomalies, i.e. the inter-technology anomalies, which arises in a set of policies of multiple security technologies. We provided also a formal model able to detect these new types of anomalies. One of the results achieved by applying the inter-technology analysis to the communication protection policies was to categorize twelve new types of anomalies. The second result of this activity was derived from an empirical assessment that proved the practical significance of detecting such new anomalies.

The second contribution of this thesis was the definition of a newly-defined type of policy analysis, named inter-domain analysis, which identifies any anomaly that may arise among different policy domains. We improved the state of the art by proposing a possible model to detect the inter-domain anomalies, which is a generalization of the aforementioned inter-technology model. In particular, we defined the Unified Model for Policy Analysis (UMPA) to perform the inter-domain analysis by extending the analysis model applied for a single policy domain to comprehensive analysis of anomalies among many policy domains. The result of this last part of our dissertation was to improve the effectiveness of the analysis process. Thanks to the inter-domain analysis, indeed, administrators can detect in a simple and customizable way a greater set of anomalies than the sets they could detect by running individually any other model.

Contents

List of Figures

List of Tables

Listings

Nomenclature

List of Abbreviations

ACL Access Control List

ARM Association Rule Mining

BDD Binary Decision Diagram

CLI Command Line Interface

CPP Communication Protection Policy

DNS Domain Name System

ECA Event-Condition-Action

FDD Firewall Decision Diagram

FOL First Order Logic

GUI Grafical User Interface

ICMP Internet Control Message Protocol

IDDs Interval Decision Diagrams

IDS Intrusion Detection System

IETF Internet Engineering Task Force

IPsec Internet Protocol Security

MPLS Multi Protocol Label Switching

NAPT Network Address Translation

NAPT Network Address and Port Translation

NIDS Network Intrusion Detection Systems

NSP Network Security Policy

PFP Packet Filtering Policy

PI Policy Implementation

QoS Quality of Service

RFC Request For Comments

SAT Satisfiability

SDN Software Defined Networking

SFTP SSH File Transfer Protocol

SNMP Simple Network Management Protocol

SSL Secure Sockets Layer

TLS Transport Layer Security

UMPA Unified Model for Policy Analysis

VLAN Virtual Local Area Network

VPN Virtual Private Network

WSS/WS-Sec/WS-Security Web Services Security

List of Symbols

Communication Protection Policy Analysis

policy implementation	
$i = (s, d, t, C, S, G)$	policy implementation (PI)
s	channel source
d	channel destination
t	security technology
C	required security levels
S	traffic selectors
G	crossed gateways

auxiliary notations	
e	network entity
$\overleftarrow{S}$	reverse list of selectors
$S\|_{f_1 \times f_2 \times \ldots}$	restrict the selector space
G^*	crossed gateways with end-points
$\overline{G}$	crossed gateways in reverse order
$P^{a,b}$	path from a to b

relationships	
$=$	equivalence
$\succ$	dominance
$\sim$	kinship
$\perp$	disjointness
$\not\perp$	not-disjointness (i.e. $\neq$, $\not\succ$, $\not\sim$)

auxiliary functions

$\mathcal{N}(i)$	node where the PI i is deployed
$\mathcal{T}(e)$	technologies supported by node e
$\mathcal{C}_{max}(i)$	maximum coefficients supported by the PI i
$\mathcal{C}_{min}(i)$	minimum coefficients acceptable for the PI i
$\pi(i)$	priority of the PI i
$\mathcal{F}_e(i)$	filtering of node e for the PI i
$\mathcal{T}^{(2)}(e)$	technologies supported at level 2 for the node e

Unified Model for Policy Analysis

n_{ik}	network field
a_{ik}	policy action
$pi_i = (n_{i1}, ..., n_{im}, a_{i1}, ..., a_{im})$	policy implementation (PI)
$\mathfrak{R}$	relations set
$n_{iq}\ \mathfrak{R}\ n_{jq}\ \wedge ... \wedge\ a_{ik}\ \mathfrak{R}\ a_{jh}\ \Rightarrow A(pi_i, pi_j)$	a generic form of a detection rule

Chapter 1

Introduction

Nowadays, computers and network communications have a pervasive presence in all our daily activities. Their correct configuration in terms of security is becoming more and more complex due to the growing number and variety of network services offered to end-users.

Security administrators, thus, have to face with this hard job, which requires very specific skills and a high level of competence. The specification of security configurations requires the definition of several technical details among several alternatives, such as security proprieties (e.g. confidentiality and data integrity), security protocols, cipher-suites, and timeouts.

Due to the complexity of the configuration security controls (e.g. firewall, VPN-gateway, authentication manager), several studies have confirmed, in fact, that many security breaches and breakdowns are attributable to administrators' responsibilities.

Wool showed that most of the firewalls he analyzed contained several problematic policies, such as very lax rules [1]:

"*Firewalls are (still) poorly configured, and a rule set's complexity is (still) positively correlated with the number of detected configuration errors.*".

As with years past, the 2014, 2015 and 2016 Data Breach Investigations Report [2–4] states that administrators were the prime actors in security incidents. As Fig. 1.1 show over 60% of incidents depend on human errors. When looking for the causes of security problems, a US government funded

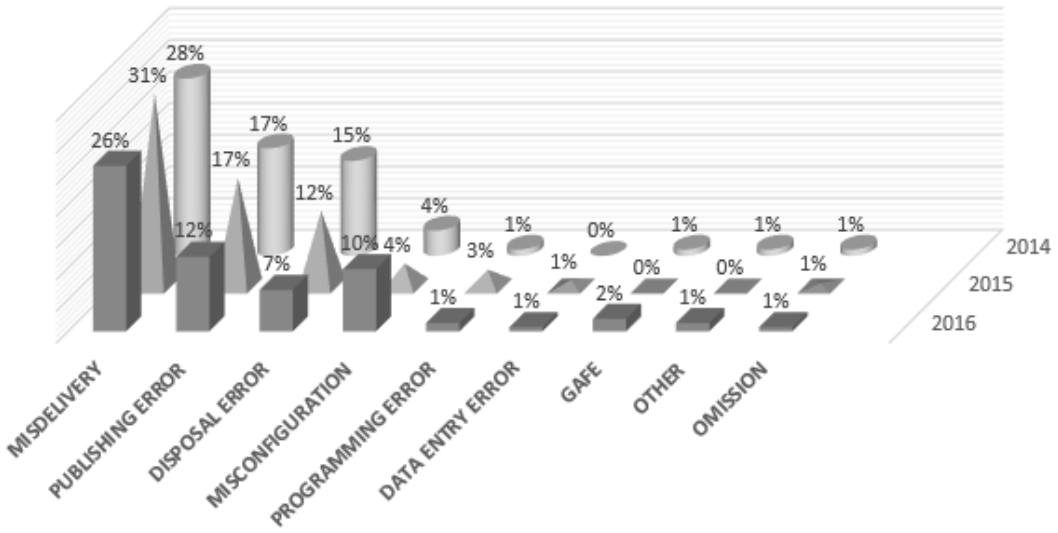


Fig. 1.1 Percentage of errors.

study reported also that the skills and competence of security administrators had strongly decreased [5].

In the last decades, the typical approach has been trial-and-error. When one or more misconfigurations are reported, the administrators corrected them by creating ad-hoc rules and repeat the process until (hopefully) no more errors are present. This methodology, although simple, is a temporary palliative since it can produce serious maintenance problems in the future. Guaranteeing the absence of misconfigurations is however nearly impossible without an appropriate software tool. It is therefore highly desirable to have a practical solution to evaluate the security configurations actually enforced which is based on sound theoretical foundations.

In order to simplify the administrator work, a different network and security configuration approach based on *policy management* was suggested in literature in [6]. The policy-based network management paradigm proposes administrators to define the security requirements by means of a set of business-level statements, namely *policy*, that are later manually or semi-automatically refined into low-level configurations for the available security controls.

From the point of view of the administrators, the introduction of a policy-based approach has simplified network management, but it has introduced other possibilities of faults and errors to take into account. An ambiguous policy

specification, in fact, may bring to an anomalous network configuration, which, in turn, may result in serious breaches and network vulnerability (e.g. blocking legitimate traffic, permitting unwanted traffic or sending insecure data). Thus, we are interested in investigating on novel approaches of anomaly analysis in network policies.

In this thesis, we define an *anomaly* as any incorrect policy specification that an administrator may introduce in the network. In particular, we indicate as policy anomaly any conflict, error or sub-optimization that may arise in the policy specification phase. Examples of anomalies are when two triggered policy rules enforce contradictory actions, or when a policy cannot be enforced due to the device capabilities (e.g. the security administrators chooses a technology not supported by an end-point or a security level too high to be enforced by the available cipher-suites), or also if the administrator defines redundant policies.

The literature has already addressed the anomaly analysis among network policies by proposing many works. However, one of the major limitations of such proposals is that each solution focuses on a specific *policy domain*. Example of domains where a policy-based management can be applied are communication protection [7], filtering [8], service function chaining [9] and others.

Another limitation is that such works have not been ported into the real activities of a network administrator[1], as well as the major part of them generally focuses on a single network security technology.

Nevertheless, the complexity of real systems is not self-contained, these approaches overlook: *(i)* the effects of the overlapping of multiple protection techniques; *(ii)* the behaviour that each network function may affect respect to other functions.

Hence it could be useful an approach of analysis and detection that gives an overview of the network errors and conflicts deriving from the wrong policy specification. For example, having a dashboard containing any irregular network conditions and events detected by the analysis approach and that an administrator wants to monitor and to be alerted (i.e. no errors and conflicts) could make more flexible and efficient the network configuration task.

[1]The only notable exception is the detection of the packet filter anomalies classified by Al-Shaer [8], which is available in some Cisco routers [10].

Therefore, in this PhD thesis we present:

1. a complete analysis of issues related to the network security policies to prevent the presence of anomalies at run-time;
2. novel classes of policy analysis, i.e. Inter-technology and Inter-domain, which aim at covering a more comprehensive set of anomalies; such newly-introduced classes help in discovering twelve new types of anomalies, whose presence in the networks has been proved by means of an empirical assessment;
3. the definition of a unified model for policy analysis, which embraces many types of analysis into a single one. This unification improves the performance of the analysis process, because administrators will be able to detect a greater set of anomalies than the sets they could detect by running individually any other model.

1.1 Thesis organization

Most of the work presented in this thesis is unpublished, and it aims at extending our previous works. In particular, we have divided this dissertation in four parts:

1. *Network Security Policy*
   The first part of this disclosure aims at providing a complete overview of the network policy world, which helps the readers in understanding the current literature on this field and how we improve it. Part of the work presented in this first part is partially described in:
   - *So you want to write a paper on policy analysis?*. Submitted to: ACM Computing Surveys;
   - *Distributed Security Policy Analysis*, the PhD thesis of Chistian Pitscheider.

   We have divided the first part of the thesis in to three chapters:

Chapter 2: provides a background on the network policies, by presenting also the nomenclature of the fundamental concepts associated to the network policies, such as policy rules and anomalies, and also a taxonomy of the main types of policies;

Chapter 3: presents the current state of the art related to policy analysis with a special focus on its main applications, which are anomaly analysis, reachability analysis and comparison analysis;

Chapter 4: describes how our work improves the current state of the art related to network policy analysis, presenting our contributions to the anomaly analysis of security policies;

2. *Communication Protection Policies*
   In this second part, we apply one of the new classes of anomaly analysis that we have defined to an already existing type of policy, i.e. the Inter-technology analysis to the Communication Protection Policies. Part of this work has been presented in the following papers:

   - *Inter-technology conflict analysis for communication protection policies*. In: CRiSIS-2014: 9th International Conference on Risks and Security of Internet and Systems, Trento (Italy), 27-29 August 2014. pp. 148-163. doi:10.1007/978-3-319-17127-2.
   - *Classification and analysis of communication protection policy anomalies*. In: IEEE/ACM Transactions on Networking, doi:10.1109/TNET.2017.2708096.

   The second part of this thesis is composed of three chapters:

Chapter 5: aims at presenting the formal model used to perform the anomaly analysis in the context of the communication protection policies. Here, we introduce the formal structures of the policies and the formulas to define and identify the anomalies against a set of policies;

Chapter 6: introduces a new graphical notation for reporting in a more intuitive way an anomaly that arises among a set of communication protection policies;

Chapter 7: concludes the second part of the thesis by showing the complexity and performance of the presented approach and the results of an empirical study carried out to prove the presence of anomalies in the context of communication protection policies;

3. *Unified Model for Policy Analysis*
   The last part of this disclosure aims at unifying all of the approaches and models of policies analysis that have been proposed in the literature or in this thesis. In particular, this part of our dissertation takes its fundamentals and is partially present in the following works:

   - *A Formal Model of Network Policy Analysis.* In: RTSI 2015 - First International Forum on Research and Technologies for Society and Industry, Torino, Italy, 16-18 September 2015. pp. 516-522. doi:10.1109/RTSI.2015.7325150
   - *Inter-function anomaly analysis for correct SDN/NFV deployment.* In: INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT, 2016, 26, 1, pp. 25-43. doi:10.1002/nem.1917

   We have divided this last part in three chapters:

   Chapter 8: presents a novel model of policy analysis that unifies the existing one (i.e. the Unified Model for Policy Analysis - UMPA), by describing its formal structure and characteristics;

   Chapter 9: shows how we applied the UMPA model into three case studies (i.e. filtering, communication protection and traffic flow policy), in order to validate the correctness and usefulness of our approach;

4. *Summation on Network Security Policies Analysis*
   We conclude this disclosure with **Chapter** 10, which summarizes the accomplishments of this work and presents possible directions to undertake as future works for each work we presented.

Network Security Policy

Chapter 2

Network Security Policy: Definitions

The first definition of policy was introduced by D. Clark and D. Wilson in [11]:

"A security policy specify the security goals that the system must meet and the threats it must resist. For example, the high-level security goals most often specify that the system should prevent unauthorized disclosure or theft of information, should prevent unauthorized modification of information, and should prevent denial of service".

In the same period another definition of policy was presented by D.C. Robinson and M.S. Sloman in [12]:

"A management policy defines the set of rules for achieving these objectives. For example, access control policy is the set of rules defining the resources a user can access and fault management policy defines where a fault should be reported and any recovery action. The policy is defined by the system administrators. "

Twelve years later, the IETF (Internet Engineering Task Force) standardized these two definitions of policy in the RFC-3198 [13] (Terminology for Policy-Based Management):

"Policy can be defined from two perspectives: (i) a definite goal, course or method of action to guide and determine present and future decisions. Policies are implemented or executed within a particular context (such as policies defined

within a business unit); (ii) policies as a set of rules to administer, manage, and control access to network resources [14]".

The RFC-3198 also provides other definitions and explanations of terms related to network security policy. We report the most relevant ones for this work:

policy rule: a basic building block of a policy-based system. A policy rule is assumed to be in ECA form (Event-Condition-Action). The fundamental construct of ECA is reactive rule in the form: "**On** *Event* **If** *Condition* **Do** *Action*", which means: "when *Event* occurs, if *Condition* is verified, then execute *Action*".

As Fig. 2.1 shown the clauses of Conditions and Actions associated with a policy rule are modelled, respectively, with aggregations of the classes Policy Event, Policy Condition and Policy Action.

policy event: is defined as any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. It is used to determine whether the Condition clause of Policy Rule can be evaluated or not. Examples of an Event include time and user actions (e.g. logon, logoff, add, update, delete).

policy condition: a representation of the necessary state and/or prerequisites that define whether a policy rule's actions should be performed. When the policy condition(s) associated with a policy rule evaluate to TRUE, then (subject to other considerations such as rule priorities and decision strategies) the rule may be enforced.

network fields: represent the possible values of the corresponding fields in a packet that matches this rule. Examples of such network fields (but are not limited to) could be the packet headers (e.g. source and destination IP addresses, the port numbers, the MIME type). In fact, other information (e.g. network node ID, traffic label, cipher algorithm etc...) could be needed to designate the events and conditions to manage.

policy action: definition of what is to be done to enforce a policy rule, when the conditions of the rule are met. Policy actions may result in the

execution of one or more operations to affect and/or configure network traffic and network resources.

policy abstraction: policy can be represented at different levels, ranging from business goals to device-specific configuration parameters. Translation between different levels of "abstraction" may require information other than policy, such as network and host parameter configuration and capabilities.

security control: are appliances or software modules within a network. They implement the functionalities needed to enforce a network security policy. Security controls can inspect the network traffic and block certain packets or modify it by changing header information of certain packets. As an example, packet filters, stateful firewalls, and application-level firewalls are used to control the traffic, whereas IPsec gateways, Virtual Private Network (VPN) terminators, and NAT/NAPT devices are able to modify the traffic.

policy refinement: is the process to determine the resources needed to satisfy policy requirements, to translate high-level policies into low-level configurations that may be enforced by the system, to verify that the set of lower level policies or configurations actually meets the requirements of the high-level policy.

policy group: is a class representing a container, aggregating either policy rules or other policy groups. It allows the grouping of rules into a Policy.

As reported in the survey [15], many proposals have been applied to analyse different types of policies: confidentiality, integrity, role base access control, military security, management and networking. In this thesis, we focused on one of this policy domain, that is the *network security domain*. We state for *Network Security Policy (NSP)* a special kind of policy that focuses on security aspects of a network.

Many types of NSPs exist and we present them in the next section (Section 2.1).

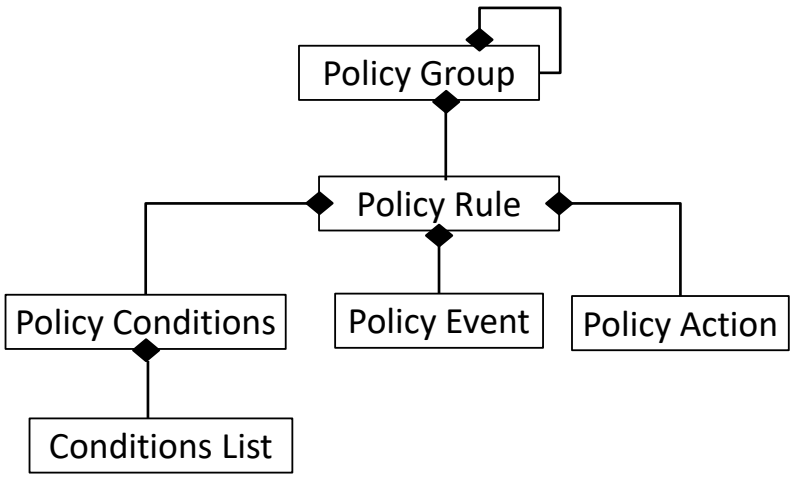


Fig. 2.1 Policy Rule Structure.

2.1 Policy types

In the domain of network security policies, there are many security controls that may drop, alter, or direct traffic. As shown in Fig. 2.2, we have considered in our analysis three main types of policies, which are forward, transformation and monitoring.

The *forwarding policies* permit to limit or modify the traversal of traffic across the boundaries of a network based on filtering and forwarding rules. We consider as forwarding policies the filtering and traffic flow policies.

The *transformation policies* specify what type of traffic should be modified and the nature of changes.

The *monitoring policies* specify wide range of traffic tasks and events to analyse.

The following sections give a brief description of each policy type, while more technical and specific details are reported in Chapter 5 and Chapter 9.

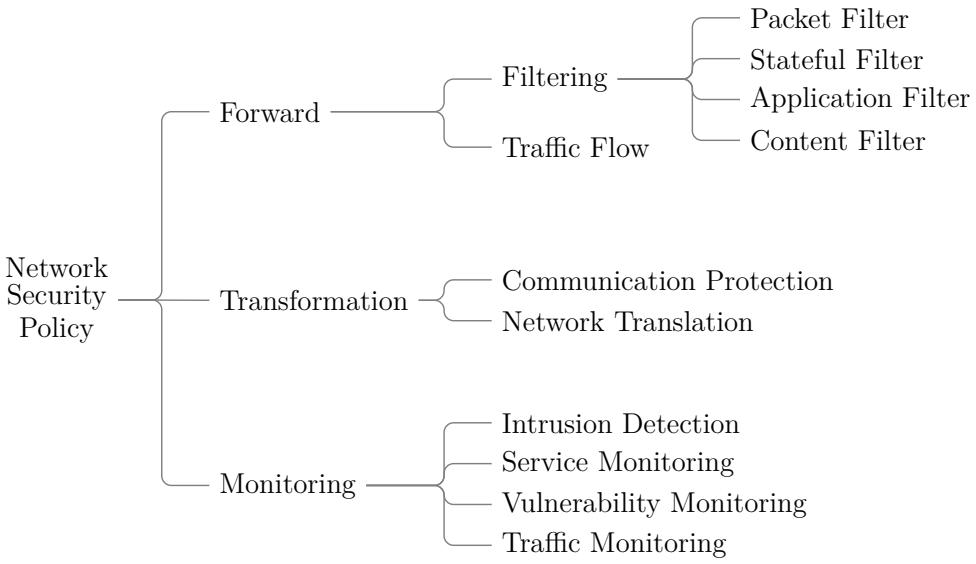


Fig. 2.2 Classification of network security policy types.

2.1.1 Filtering Policy

In our analysis, we consider four categories of filtering policies: *packet filter*, *stateful filter*, *application filter* and *content filter*.

Each category of filtering policy differs from the other ones for the considered network fields. The filtering actions, instead, are either to *accept*, which permits the packet to be delivered into or from the secure network, or to *deny*, which causes the packet to be blocked.

Packet Filter A packet filter rule is typically composed of five network fields [16]: protocol type, source and destination IP addresses, source and destination ports. Its common format of rules in a firewall policy is the followings:

<order> <src_IP> <src_Port>
<dst_IP> <dst_Port> <protocol> <action>

The rule order determines its position relative to other filtering rules. IP addresses can be a network peer (e.g. 140.192.37.120), or a network address (e.g. 140.192.37.*). Ports can be either a specific port number, a range or placeholder,

	order	src_IP	src_Port	dst_IP	dst_Port	protocol	action
R_1	1	192.168.1.*	*	*	80	*	deny
R_2	2	192.168.1.*	0-1024	192.168.3.*	0-1024	TCP	allow
R_3	3	192.168.1.*	0-1024	192.168.2.*	22	*	allow
R_4	4	192.168.1.1	*	192.168.2.*	*	UDP	deny
R_5	5	192.168.*.*	*	192.168.*.*	*	*	allow

Table 2.1 Example of packet filter rules.

indicated by "any" or "*", to represent all port number. An extract of a packet filterer policy is shown in Table 2.1.

In same cases, the packet filtering condition set supports time-based conditions [14] and its structure can be extended with a further field: Its common format of rules in a firewall policy is the followings:

<order> <src_IP> <src_Port> <dst_IP>
<dst_Port> <protocol> <time> <action>

Stateful Filtering The stateful filtering improves the packet filter functionality by also maintaining connection states at the transport layer, and it analyses the informations in DNS and ICMP messages. The Stateful firewall gives origin to stateful firewalls (or stateful packet filter) [17]:

"*If a firewall decides the fate of every packet solely by examining the packet itself, then the firewall is called a* ***stateless firewall****. If a firewall decides the fate of some packets not only by examining the packet itself but also by examining the packets that the firewall has accepted previously, then the firewall is called a* ***stateful firewall*** [18]".

The stateful firewalls analyse the transport headers, and maintain a state table that associates a set of Boolean variables to each connection (*frag, tcp_flg, tcp_opt, syn, tcp_state*). In addition to the filtering action, the stateful firewalls perform the action *REJECT*. The REJECT action works basically the same as the DROP, but it also sends back an error message to the host that sent the rejected packet.

Application Filtering The application firewalls have the ability to (recursively) extract nested information in TCP or UDP packets and take decisions using application protocol data and states [19]. The most widespread specialized firewall is the Web Application Firewall (WAF), which deeply analyzes HTTP traffic. Typically an application firewall supports: *(i)* regular expressions to define conditions on text fields, like URL (i.e. the domain filtering like path and domain conditions on URL) and MIME type; *(ii)* application-proxy gateways action to keep track of authenticated users and to limit the maximum number of simultaneous users/connections per user.

Content Filtering Content filters restrict or block the access to resources (like web or mail resources) that are deemed objectionable (such as drugs, profanities, hate, pornography, etc...) [20]. Content filters usually works by specifying regular expressions that, if matched, indicate undesirable content that is to be screened out. Some content filters block images from websites and webmails, by analysing the graphical content of an image, and block all suspicious images, so that a blank or checkered box, is displayed in place of the blocked image. Content filter and the products that offer this service can be divided into Web filtering, which is the screening of Web sites or pages, and e-mail filtering, winch is the screening of E-mail for spam or other objectionable content.

2.1.2 Traffic Flow Policy

Routing security controls implement a packet forwarding on a static routing table or a dynamic routing protocol (i.e. SDN-Software Defined Networking [21]).

Recently, traffic flow policies are typically used to split an entire network into multiple logical (virtual) networks, called flow entries, that can be seen as individual networks.

OpenFlow is an example of new technology used to add, update, and delete flow entries, both reactively (in response to packets) and pro-actively [22].

The network fields, shown in Table 2.2, supported by OpenFlow are:

Ingress port: the numerical representation of incoming port, starting at 1. This may be a physical or switch-defined virtual port;

Ether src the Ethernet source address;

Ether dst the Ethernet destination address;

Ether type: the Ethernet type of the OpenFlow packet payload, after VLAN tags;

VLAN id: the VLAN identifier of outermost VLAN tag;

VLAN priority: the VLAN PCP field of outermost VLAN tag;

MPLS label the MPLS label value in the MPLS header;

MPLS EXP traffic class: the MPLS experimental value[1] in the MPLS header;

IPv4 src: the IPv4 source address (can use subnet mask or arbitrary bitmask);

IPv4 dst: the IPv4 destination address (can use subnet mask or arbitrary bitmask);

IPv4 proto: the IPv4 protocol type;

IPv4 src port: the IPv4 source port number;

IPv4 dst port: the IPv4 destination port number;

IPv4 ToS bits: the IPv4 type of service;

ICMP type: the ICMP type value;

ICMP code: the ICMP code value;

[1]Used fot QoS marking; the field is no longer used for truly experimental purposes.

Ingress port	Ethernet src	Ethernet dst	Ethernet type	Vlan id	Vlan priority	MPLS label	MPLS EXP	IPv4 src	IPv4 dst	IPv4 protocol	IPv4 src port	IPv4 dst port	IPv4 ToS bits	ICMP type	ICMP code

Table 2.2 Fields from packets used to match against flow entries.

2.1.3 Communication Protection Policy

Communication protection policies (CPPs) originate from legal (e.g. the EU privacy law [23]) and business security requirements, because they are used to specify how to protect network communications. Their correct deployment is then crucial in several areas, such as protection of intellectual properties, and confidentiality of financial or corporate data. In some cases, such as companies that host services or provide cloud-based resources, CPPs may be extremely complex and articulated.

Communication protection policies can be enforced by using different security protocols, like IPsec [24], TLS [25], SSH [26], WS-Security [27] and S-FTP [28]. Due to this, CPPs have to include several types of parameters, which generally are cypher suites, time-outs, end-points, traffic types to be protected, and type of tunnels (i.e. end-to-end channels or site-to-site).

In this paragraph we do not provide further details about the CPPs and their formalism, because they are presented in depth in Chapter 5, while examples of low-level configurations to implement CPPs are provided in Appendix B.1

2.1.4 Network Translation Policy

Translation devices include Network Address Translation (NAT) and Network Address and Port Translation (NAPT) controls, they modify the packet header according to a policy.

NAT changes the IP addresses and NAPT changes IP address and ports [29].

2.1.5 Monitoring Policy

The monitoring policies are mainly divided in four main groups of policy types: *intrusion detection*, *service monitoring*, *vulnerability monitoring* and *traffic monitoring*.

In all monitoring policies, the policy conditions are assessed to a set of network/security measurements, while the actions consist in solving, limiting or alerting suspicious behaviour.

Network and security measurements are roughly divided into two groups that are passive and active measurement. Passive measurement measure network traffic by observation, without injecting additional traffic in the form of probe packets. The advantage of a passive measurement is that it does not generate additional network overhead, and thus does not influence network performance. Unfortunately, passive measurements rely on installing in-network traffic monitors, which is not feasible for all networks and require large investments. Active measurements, on the other hand, inject additional packets into the network, monitoring their behaviour For example, the popular application "ping" uses ICMP packets to reliably determine end-to-end connection status and compute a path's round-trip time.

Chapter 3

Network Security Policy: Analysis

In this thesis, we refer *policy analysis* as the general process that analyses and checks the violation of some properties against a set of policies. Many possible applications of this process have been identified in the literature. In particular, we divided the policy analysis in Anomaly analysis, Reachability analysis and Policy comparison.

The *Anomaly Analysis* searches for any incorrect policy specification that an administrator may introduce in the network, which we indicate as *anomalies*. It checks any potential error, conflict and sub-optimization within a single or a set of security policies.

Instead the *Reachability Analysis* evaluates allowed communications within a computer network. It can determine if a certain host can reach a service or a set of services. In general, reachability analysis is performed on-line by using tools such as "ping" or "traceroute". By using an accurate representation of the network and its security policies, reachability analysis can also be performed off-line, on an abstract network and policy representation, during the design phase.

The last is the *Policy Comparison*, which compares two or more network security policies and represents the differences between them in an intuitive way. In this thesis, a network security policy includes a single or a set of concrete security control configurations, or high-level security requirements

to be enforced in the whole network. One of the best use-cases of policy comparison is to verify that a desired network security policy is implemented correctly by comparing the designed high-level policy with the concrete network configuration. In addition the policy comparison is used to detect difference with previous version in case of (dynamic) changes.

The rest of the section presents the state of art for each type of analysis. We also provide in Appendix A some concepts and definitions about the mathematical models exploited in the Policy Analysis approaches presented in the literature.

3.1 Anomaly analysis

A policy anomaly, typically, occurs when a set of policies (two or more) are simultaneously satisfied. This implies that the combined actions of the policies may produce different results depending on the order of execution of these actions.

In this thesis, we distinguish three types of policy anomaly, which are:

- **Policy conflict**: it arises when the effect of one security policy is influenced or altered by another one, e.g. the actions of two rules (that are both satisfied simultaneously) contradict each other. The policy systems must provide conflict detection and avoidance or resolution mechanisms to prevent this situation;
- **Policy error:** it occurs when the attempts to enforce the policy actions fail, either due to temporary state or permanent mismatch between the policy actions and the device enforcement capabilities;
- **Policy sub-optimization:** when redundant rules or other more efficient policy implementations (in terms of resources and security) are present.

Policy anomaly can often occur when there are multiple authors defining policies for a given system (network and security administrators). The creation or modification of a policy is a difficult task, because a new/updated policy can affect the behaviour of existing policies, defined by other people in different

times. Therefore, the correct refinement of network policies is challenging since an administrator must infer and choose several technical details among several alternatives. For example, in case of a security requirement wanting to enforce secure communications, an administrator should select the security protocol, the cipher-suite, the timeouts, and other parameters to apply.

Then, we need to apply an anomaly analysis process in order to look for any anomaly (i.e. conflict, error and sub-optimization) within a set of policy rules. Up to now, the literature has applied the anomaly analysis to a single policy or to a set of policies of interconnected security controls. Those methodologies are known in the literature respectively as the *Intra-policy* and *Inter-policy* analysis, of which we provide further details in the next sections.

3.1.1 State of the Art

In literature, anomaly analysis is mainly applied to single types of security controls and there is no complete solution that analyses all types of security controls. In particular, research is mainly concentrated on Intra- and Inter-policy analysis of packet filter and IPsec configurations.

Packet Filtering Qian *et al.* [30] propose a framework to automate Access Control List (ACL) analysis, capable of detecting and removing redundant rules. Furthermore, the proposed algorithm is also capable of discovering and repairing inconsistent rules, of merging overlapping or adjacent rules, and rewriting ACLs to be more readable. This was one of the first proposals that aims at resolving policy misconfigurations. The limitation of this work is that it considers only few types of policy anomalies.

The anomaly analysis of packet filtering policy was firstly introduced by Al-Shaer and Hamed [16]. The authors present a classification scheme for packet filter rule relations, based on which they defined the four types of Intra-policy rule anomalies (shadowing, correlation, generalization and redundancy).

"Two rules are shadowed when they enforce different actions and both rules match the same packets. Two rules are correlated when they enforce different actions and both rules have some matching packets in common. A rule is a generalization of a second rule when they enforce different actions and the

second rule matches the same packets as the first one but not vice versa. Two rules are redundant when they enforce the same action and match the same packets."

Then, Al-Shaer *et al.* introduce an extension of the Intra-policy classification analysis base, called Inter-policy rule anomalies, in the extension of the first paper [8]. Inter-policy analysis evaluates rule relations between serially-connected packet filters. Al-Shaer *et al.* define five new Intra-policy anomalies (shadowing, spuriousness, redundancy, correlation and irrelevance).

"Two rules from two different firewalls are shadowed when they match the same packets and the rule from the first firewall blocks a packet that is permitted by the second rule. Two rules from two different firewalls are spurious when they match the same packets and the rule from the first firewall permits the packet which is blocked by the second rule. Two rules from two different firewalls are redundant when they match the same packets and both rules block the packet. Two rules from two different firewalls are correlated when they have some matching packets in common and enforce different actions. A rule is classified as irrelevant if there is no possible traffic which can be matched by the rule, for example the source and destination address belong to the same zone."

Based on the work of Al-Shaer *et al.*, other researchers proposed alternative models and classification schemas. These works prove that Al-Shaer's classification scheme is valid and can be applied to real world scenarios. The main limitation of all these approaches is that they cannot handle other security controls but packet filters.

Firecrocodile [31], propose by Lehmann *et al.*, was the first approach to help network administrators to correctly configure PIX firewalls. The tool builds a model which represents the PIX configuration file and performs the analysis on it. In addition to anomaly analysis, they verify also the configuration file for policy violations. Its main limitation is that it can analyse only Intra-policy packet filtering rules of Cisco PIX configurations.

FIREMAN [32], propose by Yuan *et al.*, uses binary decision diagrams (BDDs) to represent packet filtering policies. In addition to an Intra-policy analysis, it also verifies that an end-to-end policy is correctly implemented by the filtering configurations. The model is designed for packet filters only and does not support any other type of security control.

Jeffrey *et al.* [33] propose to use a SAT-solver instead of BDDs for firewall analysis. They claim that the problem can be represented as SAT and does not require BDDs. The reduced complexity of the problem has a real advantage in terms of performance. In the performance evaluation the authors also reimplement FIREMAN with a SAT-solver and show that it is more efficient.

Ferraresi *et al.* [34] extend Al-Shaer's classification and propose two automatic anomaly resolution algorithms, providing a formal proof for the existence of the solution and the algorithm convergence.

Golnabi *et al.* [35] extend Al-Shaer's analysis using Association Rule Mining (ARM), a data mining technique. The anomaly detection based on the mining exposes many hidden but not detectable anomalies by analysing only the firewall policy rules, resulting in two new non-systematic misconfiguration anomalies: *Blocking existing service* and *Allowing traffic to non-existing service* anomaly. The first misconfiguration case is blocking a legitimate traffic from a trusted network to an "existing" service, while the other case of the misconfiguration permits a traffic destined to a non-existing service.

Liu *et al.* [36] focus only on detecting and removing redundant rules. The authors categorize redundant rules into upward redundant rules and downward redundant rules. Upward redundant rules are rules that are never matched, whereas downward redundant rules are rules that are matched but enforce the same action as rules with lower priority. The model presented is based on a data-structure named Firewall-Decision-Diagram (FDD).

Mohamed *et al.* [37] propose a complementary work called structured firewall design, which consists of two steps. First, one designs a firewall using a FDD instead of a sequence of possibly conflicting rules. Second, a program converts the firewall decision diagram into a compact, yet functionally equivalent, sequence of rules. This method addresses the consistency problem because a FDD is anomaly-free.

Cuppens at al. [38] propose an anomaly resolution approach in addition to the detection one. Although they support only shadowed and redundant anomalies among single policies, their resulting policy is anomaly free. Another feature of this model is that it can rewrite a policy in its positive or negative form. The positive form of a policy contains only ALLOW rules whereas the negative form only DENY rules. The authors extend their model to support

also Inter-firewall analysis [39]. Furthermore, the authors define three new anomalies (reflexivity, misconnection, and irrelevance). A reflexivity anomaly occurs when both source and destination address are within the same zone. A misconnection anomaly occurs when a rule blocks traffic that is not explicitly blocked by the most-upstream firewall. An irrelevance anomaly occurs when a firewall is not within the minimal route that connects the source zone[1].

Garcia-Alfaro *et al.* [40] propose the integration of network intrusion detection systems (NIDS). The proposed model can detect both Intra- and Inter-policy packet filter rule anomalies. The main improvement over Al-Shaer's model is that it can also handle NIDS, and not only packet filters. The tool can also verify which security controls are on the path of a given packet based on its source and destination address. This work has been later integrated into the MIRAGE tool [41].

Bouhoula *et al.* [42] suggest a different approach to this topic by using an inference system to detect Intra-policy anomalies. They use the inference system to construct a tree representation of the policy. The construction process is efficient and optimized for memory consumption. The inference contains a condition that stops the construction of a specific branch when no anomaly can be found. The resulting classification tree contains rule anomalies in its leaves. The disadvantage of this approach is that it is not able to check for Inter-policy anomalies, furthermore it is not capable of handling security policies such as IPsec/VPN.

Afterwards, Bouhoula *et al.* [43] propose an other formal approach for detecting and solving firewall misconfigurations Their approach also detects anomalies between multiple rules and not only rule pairs. Furthermore, they introduce a new classification of firewall anomalies: superfluous rule-class anomalies and conflicting rules-class anomalies. Superfluous rule-class anomalies include the shadowing and redundant anomaly and conflicting rules-class anomalies include the correlation and generalization anomaly. The new classification is used because superfluous rule-class anomalies can be resolved by their algorithm.

[1]This definition of irrelevance anomaly is different form the definition propose by Al-shaer *et al.* in [8].

Abedin *et al.* [44] present an algorithm that detects and solves any anomaly present in the policy rules. The prosed algorithm operates by reordering and splitting the existing set of rules in order to generate a new one that is anomaly-free. The new set of rules will be smaller than the original one, by increasing the overall efficiency of the firewall. This work has not been validated but has a proof of correctness of the algorithm.

Basile *et al.* [45] present an new analysis derived form the work of Al-shaer *et al.* [8]. The authors introduce a new formal model for policy specification, named Geometrical Model [46], it is based on a set of rules, a default action and a resolution strategy. The presented model can identify all types of Intra-policy anomalies defined by Al-Shear. Furthermore, the authors present two new anomaly types: general redundancy anomaly and the general shadowing anomaly. The general redundancy anomaly occurs when a rule is redundant due to the union of multiple rules. The general shadowing anomaly occurs when a rule is shadowed by the union of multiple rules.

Hu *et al.* [47], [48] propose a rule-based segmentation technique and a grid-based representation to identify policy anomalies and derive effective anomaly resolutions. They also present a proof-of-concept implementation of a visualization-based firewall policy analysis tool called Firewall Anomaly Management Environment (FAME).

Basumatary *et al.* [9] propose a formal model for firewall anomaly detection. They represent firewall rules by a topological-temporal model and use a model checker to verify the firewall policy.

Krombi *et al.* [49, 50] propose a procedure that synthesizes an automaton that implements a given security policy. They use our automaton to verify completeness, detect anomalies, and discover functional discrepancies between several implementations of a security policy.

Stateful firewalls Only recently stateful firewalls have been integrated into analysis models. One of the few examples is presented in [51] and [17]. Cuppens and Garcia-Alfaro [51] propose a solution for Intra-policy analysis of stateful firewalls. With the introduction of stateful firewalls they also present new types of anomalies classes (Intra-state and Inter-state rule anomalies). Intra-state rule anomalies occur only between stateful rules and beside the known

anomalies from the stateless analysis, include two new anomaly types. The first new anomaly arises when the firewall blocks packets during the three-way handshake. The second new anomaly arises when the firewall blocks packets during the connection termination. Inter-state rule anomalies occur between stateful and stateless rules when application layer protocols establish multiple connections and at least one of these connections is blocked, an example of such a protocol is FTP. The proposed algorithmic solution to handle and eliminate such types of anomalies is based on a general automata describing the stateful rules. This initial work has been completed and formalized in [17]. Although the introduction of stateful firewall into the analysis process was a very important step, both solutions are still missing the Inter-policy analysis.

Application Firewall Basile *et al.* [52] present an extension of their Geometrical Model which can perform anomaly analysis of application-level firewall configurations. The extended model can identify all policy anomalies introduced in their previous work. The main contribution of this improvement is the anomaly analysis of firewall rules including regular expressions. The model transforms the regular expressions into deterministic automata and calculates rule intersection based on them.

Traffic flow policies In the last years, several works have been proposed to find anomaly misconfigurations in OpenFlow, in order to allow multiple applications to run on the same physical network in a non-conflicting manner. Al-Shaer and Al-Haj present a tool, FlowChecker, to identify any Intra-switch misconfiguration within a single FlowTable [53]. They also describe the Inter-switch or Inter-federated inconsistencies in a path of OpenFlow switches across the same or different OpenFlow infrastructures.

NICE, proposed by Canini *et al.* [54], performs a symbolic execution of OpenFlow applications and applies a model checking technique to explore the state space of an entire OpenFlow network.

Finally, Batista *et al.* present an approach for conflict detection using several first-order logic rules to define possible antagonisms and employ an inference engine to detect conflicting flows before the OpenFlow controller [55].

Communication protection policies The current literature contains several works about anomaly detection in CPPs, however, the research in this area is solely focused on IPsec, and overlooks the effects of multiple overlapping protection techniques.

Zao [56] introduce an approach based on the combination of conditions that belong to different IPsec fields. The same idea was used by Fu *et al.* [7] to describe a number of conflicts between IPsec tunnels, discovered through a simulation process that reports security requirements violations. In their analysis, the policy anomalies are identified by checking the IPsec configurations against the desired policies written in a natural language. In practice, an anomaly occurs when the policy implementations do not satisfy the desired policies. In addition, Fu *et al.* propose a resolution process that finds alternative configurations to satisfy the desired policy.

Al-Shaer *et al.* analyze the effects of IPsec rules on the protection of networks [57], by proposing a number of ad-hoc algorithms and formulas to detect these problems. They formalized the classification scheme of [7] and proposed a model based on OBDD (Ordered Binary Decision Diagrams) that not only incorporates the encryption capabilities of IPsec, but also its packet filter capabilities. They also identify two new IPsec problems (channel-overlapping and multi-transform anomalies). The first one occurs when multiple IPsec sessions are established and the second session redirects the traffic of the first one (similar to the case depicted in Fig. 5.4). On the other hand, the multi-transform anomalies occur when a data protection is applied to an already encapsulated IPsec traffic and the second protection is weaker than the first one. The same authors also describe a classification system for conflicts between filtering and communication protection policies [58].

Niksefat *et al.* [59] present a two improvements over the Al-Shaer's solution [57], a faster detection algorithm and the possibility to resolve the detected anomalies.

Finally, Li *et al.* classify IPsec rules in two classes: access control lists (ACL) and encryption lists (EL) [60].

3.1.2 Summary of State of the Art and Research Directions

As shown in Table 3.1, there are may works on anomaly analysis of filtering policies. For the most part they have been published between 2005 and 2008 and are related to packet filter. This area of anomaly analysis has been extensively analysed and almost all of the works include an implementation and are validated in a real scenario or by theorem proofs. Although the definition of Inter-policy anomalies was defined in 2005, only few works support it. Despite this last shortcoming, in our view the research on anomaly analysis of filtering policies has reached maturity. Otherwise only recently anomalies in stateful and application firewall policy have been analysed. We expect that these are two hot topics in filtering policy anomaly analysis. The research is only at the beginning and there can be introduced much more improvements.

		Data		Policy Type			Features		Validation			
	Y	AUT	CIT	PKF	SFF	APF	INT-PLC	RSL	PRT	TST	EMP	THEO
[16]	2003	Al-Shaer	91	✓								
[8]	2005	Al-Shaer	141	✓			✓		✓	✓	✓	
[35]	2006	Al-Shaer	41	✓					✓	✓	✓	
[31]	2006	Lehmann	2	✓					✓	✓	✓	
[32]	2006	Yuan	121	✓			✓		✓	✓		
[33]	2009	Jeffrey	29	✓					✓	✓		
[34]	2007	Ferraresi	12					✓	✓	✓		✓
[36]	2005	Liu	30	✓				✓				
[37]	2007	Liu	65	✓				✓				
[38]	2005	Cuppens	47	✓				✓				✓
[39]	2006	Cuppens	8	✓				✓	✓	✓		✓
[40]	2007	Cuppens	56	✓				✓	✓	✓		
[41]	2010	Cuppens	12	✓				✓	✓	✓		
[51]	2012	Cuppens	8	✓	✓			✓	✓	✓		
[17]	2013	Cuppens	7	✓	✓			✓	✓	✓		
[44]	2006	Abedin	18	✓				✓				✓
[42]	18	Bouhoula	18	✓				✓	✓	✓		
[43]	2014	Bouhoula	7	✓			✓	✓	✓	✓		
[46]	2008	Basile	5	✓			✓		✓	✓	✓	
[45]	2012	Basile	12	✓			✓		✓	✓	✓	
[52]	2014	Basile	6	✓		✓	✓		✓	✓	✓	
[47]	2010	Hu	6	✓			✓		✓	✓	✓	
[48]	2012	Hu	49	✓			✓		✓	✓	✓	
[49]	2014	Krombi	2	✓				✓				✓
[50]	2015	Krombi	1	✓				✓				✓

Y= Year; AUT= Author (reference author); CIT= Citation numbers (scopus and/or scholar); PKF=Packet Filter; SFF= Stateful Firewall; APF= Application Filter; INT-PLC= Inter-policy; RSL= Resolution; PRT= Prototype; TST= Test; EMP= Empirical evaluation; THE= Theoretical validation

Table 3.1 Summary of state of the art in Filtering Policy Analysis.

As shown in Table 3.2, there are much less works on anomaly analysis of communication protection policy than in to filtering policy. Those papers mainly focus on their analysis on potential incompatibilities and anomaly among IPsec policies, such as redundancy protection anomaly or channel overlapping misconfiguration. The authors overlook the possible interaction with other policy of distinct technology (e.g. TLS, SSH). In our vision, the analysis of anomalies that derive from the interactions between multiple technologies is an interesting research topics.

		Data		Policy Type		Features		Validation			
	Y	AUT	CIT	IPSec	PKF	INT-THC	RSL	PRT	TST	EMP	THEO
[7]	2001	Fu	29	✓			✓	✓			
[57]	2005	Al-Shaer	38	✓	✓			✓	✓	✓	
[58]	2006	Al-Shaer	88	✓	✓			✓		✓	
[60]	2006	LI	9	✓	✓			✓			
[59]	2010	Niksefat	6	✓				✓	✓		

Y= Year; AUT= Author (reference author); CIT= Citation numbers (scopus and/or scholar); PKF=Packet Filter; INT-THC= Inter-technology; RSL= Resolution; PRT= Prototype; TST= Test; EMP= Empirical evaluation; THE= Theoretical validation

Table 3.2 Summary of state of the art in communication protection policy analysis.

Finally, efficiency of a policy-based system is important in the same way of having the means of detecting and resolving any anomalies that arise. Unfortunately only a few works of filtering and communication protection policy propose also anomaly resolution techniques in addition to the anomaly analysis.

3.2 Reachability analysis

Reachability analysis evaluates allowed communications within a computer network. Furthermore, it can determine if a certain host can reach a service or a set of services.

"*Quantifying and querying network reachability is important for security monitoring and auditing as well as many aspects of network management such as troubleshooting, maintenance, and design*" [61].

Reachability analysis can be performed both online and offline. Online reachability analysis is performed on a deployed system by injecting test packets

and verifying on different points of the network that those packets are present. Off-line reachability analysis is performed on a model of the system without direct interaction with a real network.

Online reachability analysis

In the on-line reachability analysis, the correctness is guaranteed because real packets are injected into the network and are processed by all devices on the path. The disadvantage is that special probes have to be installed within the network. Inserting these probes for the most use-cases is difficult and involves a precise planning and execution. Due to this limitation, on-line reachability is mainly used to verify the correct behaviour of single network components and not for validating the entire network.

Offline reachability analysis

The off-line reachability analysis has the advantage that the system to be analysed does not need to be deployed. This means that it can be used during the design and maintenance tasks. Furthermore, it can also verify reachability on alternative paths, and therefore test fault-tolerance properties of the systems.

This approach is much more dynamic and the planning and execution is much less invasive than on-line approach. The disadvantage is that the correctness of the result is based on the correctness of the model. This also implies that if some parts of the network can not be expressed with the model, the application of this type of analysis is limited or even impossible. Furthermore, all policies must be represented correctly by the instantiation of the model.

3.2.1 State of the Art

On-line reachability analysis in general is performed by using tools such as "ping", "traceroute", and "tcpdump". There are only a few publications regarding this on-line reachability analysis. The general approach taken in literature is to insert a traffic generator and a traffic analyser into the network. The most promising work is presented by El-Atawy *et al.* [62] and Al-Shaer *et al.* [63], because they propose a traffic generator analyses first the security policy and based on this analysis, the most relevant packets are generated. The limitation of these two works is that they can be applied to single firewalls only.

Brugger *et al.* [64] propose a different approach by using a test-policy that generates the effective queries. This has the advantage that queries can be written in a more abstract level and the reachability analysis is closer to a policy comparison.

Mayer, Wool, and Ziskind [65] present a firewall analysis engine named Fang. It is the first approach towards off-line reachability analysis of computer networks containing only packet filters. The proposed solution takes as input the network topology and the configuration files of the deployed packet filters. A user interface to perform reachability queries is provided and the queries are evaluated by the tool. In the extended versions of the paper [66] and [67] the query interface has been improved and the most relevant queries are generated automatically by the tool.

Marmorstein and Kearns [68] [69] [70] propose a tool named ITVal for analysing IPtable configurations. It is the first open source implementation of a firewall analyser capable of performing simple reachability queries.

Oliveira *et al.* [71] also propose an open source tool for firewall analysis named Prometheus. They present a flow connectivity analysis to verify whether a server is accessible from different locations of the network.

Eronen *et al.* [72] propose to use logic programming and a generic inference engine to analyse Cisco router access lists. Their approach has the advantage that reachability algorithms are not hard-coded. New rule analyzing functions can easily be added and the expert knowledge can be expressed in a compact form. Furthermore, their model can also analyse the rule structure, for example if a rule is never matched.

Hazelhurst [73] presents a model for reachability analysis and policy comparison. The model represents packet filter rules internally as BDDs and has a graphical user interface (GUI) for human interaction. To perform a reachability analysis the user expresses a query as a Boolean expression and the result is displayed in tabular form. The user can express queries for packets, which are allowed and denied. Furthermore, the output can be formatted to obtain different levels of abstraction.

Xie *et al.* base their reachability analysis on graph theory and dynamic programming [74]. The solution is able to calculate the upper and lower bound of reachability. The upper bound defines that there is at least one possible path for reachability and the lower bound defines that all possible paths allow reachability. The model can be used to represent static NAT, routing and filtering rules based on the destination addresses, but it does not take into account the existence of connectionless and connection-oriented protocols. Although the correctness of the model is given, it is purely theoretical and lacks experimental results. Bandhakavi *et al.* [75] present an extension to Xie's work to overcome limitations. They use a more general model to describe firewalls, packet filtering and transformation rules, thus adding the possibility to handle policies that depend on source addresses and filtering states.

Matousek et. al. [76] present a formal model for reachability analysis applied to networks containing packet filters and dynamic routing. The authors propose to represent ACLs as a First-Order Logic (FOL) function and to use Interval Decision Diagrams (IDDs) [77] to store them. IDDs are efficient for conjunction, disjunction and inclusion over this functions.

Khakpour and Liu [78–80] present a reachability analysis tool called Quarnet. Quarnet supports connectionless (stateless router/firewall and static NAPT) and connection-oriented transport protocols (stateful router/firewall and dynamic NAPT). The paper presents a model for calculating network reachability metrics and also includes a performance analysis. The solution is based on an internal representation of the network on which reachability queries are executed. The authors first calculate a Firewall Decision Diagram (FDD) to represent the global policy and afterwards compute two matrices which contain the effective reachability information needed. Although the single reachability

queries are very fast to compute, the computation of the internal network representation does not achieve similar times to the querying operations.

Nelson *et al.* [81] present a firewall analysis tool named Margrave, which is able to perform basic reachability queries.

Mai *et al.* [82] present a reachability verification algorithm based on a SAT solver, where the reachability query is represented as a Boolean formula. The SAT solver searches for a symbolic packet which can be forwarded between two vertices of the network.

Al-Shaer *et al.* [83] present a tool named ConfigChecker. The tool models the global end-to-end behaviour of a network and can perform reachability verification, identify bogus entries, verify IPsec tunnel integrity and discover backdoors.

Another theoretical approach used to compute the network-wide reachability, has been proposed by Sveda *et al.* [84]. This approach uses traditional graph-based algorithms, such as Floyd-Marshall, whereas [74] and [75] require ad-hoc techniques to mimic routing protocols. To calculate the reachability of the network these other works use the encoding problem into SAT instance solved by automatized solvers. They describe how to represent both routing and filtering devices, but do not mention how to express packet transformation rules.

Kazemian *et al.* [85] present a generalization of Xie's work [74] based on "Header space" information of packets. Their algorithm is compatible with filtering, routing, and transformation technologies. However, this approach is limited to packet filters and cannot be used for filtering and security devices which work at a higher level of the ISO/OSI stack.

3.2.2 Summary of State of the Art and Research Directions

Table 3.3 summarizes the difference in publications regarding reachability analysis. In summary all publications support packet filter, just a few of those works support also routing and NAT and only [64] support the stateful firewalls. The main technique used for reachability is off-line analysis.

Although there are different models of analysis, as concerning performance testing and validation, these are carried on just by very few works in real use cases. At the best of our knowledge, in the current literature none has faced the problem to extend reachability analysis in VPN context.

Taking into account the single limitations of each work and the common shortage of the whole literature, we think that unique model that embeds on it all the different network and security controls is needed.

		Data		Policy Type				Features		Validation			
	Y	AUT	CIT	PKF	SFF	NAT	ROU	ON	OFF	PRT	TST	EMP	THEO
[65]	2000	Mayer		✓					✓	✓			
[67]	2006	Mayer		✓					✓	✓		✓	
[66]	2001	Wool		✓					✓	✓		✓	
[72]	2001	Eronen	60	✓					✓				
[73]	2003	Hazelhurst	60	✓					✓				
[74]	2005	Xie	60	✓		✓	✓						
[68]	2005	Marmorstein	18	✓					✓				
[69]	2005	Marmorstein	8	✓		✓			✓				
[70]	2007	Marmorstein	3	✓									
[62]	2005	Al-Shaer	24	✓				✓		✓	✓		
[63]	2005	Al-Shaer	28	✓				✓		✓	✓		
[83]	2005	Al-Shaer	3	✓					✓	✓	✓		
[76]	2008	Matoušek	19	✓		✓							
[71]	2009	Oliveira	8	✓					✓	✓	✓		
[75]	2008	Bandhakavi	2	✓		✓	✓						
[78]	2009	Liu	2	✓		✓	✓		✓	✓	✓	✓	
[79]	2010	Liu	9	✓		✓	✓		✓	✓	✓	✓	
[80]	2013	Liu	11	✓		✓	✓		✓	✓	✓	✓	✓
[81]	2010	Nelson	40	✓		✓	✓		✓	✓	✓	✓	
[82]	2011	Mai	135	✓		✓	✓		✓	✓	✓	✓	
[85]	2012	Kazemian	113	✓					✓	✓	✓	✓	
[64]	2013	Brugger	2	✓	✓	✓	✓	✓		✓	✓	✓	

Y= Year; AUT= Author (reference author); CIT= Citation numbers (scopus and/or scholar); PKF=Packet Filter; SFF= Stateful Firewall; ROU= Routing; ON= On-line verification; OFF= Off-line verification; PRT= Prototype; TST= Test; EMP= Empirical evaluation; THE= Theoretical validation

Table 3.3 Summary of state of the art in Reachability Analysis.

3.3 Policy comparison

Policy comparison has different application scopes, in general it can be divided into two categories: *change impact* and *implementation verification*.

Change impact is used to verify the effective impact on a policy after inserting, removing and/or modifying a policy. Therefore, the original policy is compared with the modified one.

The implementation verification instead is used to verify that a policy implementation is correct and enforces all rules as specified in the design policy. In this scenario, the design policy for the most part is written in a high-level language.

The change impact and implementation verification can be both applied on *single* and *multiple* policy comparison. Single policy comparison is performed between two single policies one to one. This approach is limited to one single policy type, therefore a filtering policy can not be compared with a data protection policy. Multi policy comparison is performed between to complete network configurations and associated policies. In this case, different types of policies can be involved at the same time.

3.3.1 State of the Art

Guttman [86] presents the first model of policy comparison. It can generate for a given network the required access control lists based on a global policy. Furthermore, the global policy can be used as a reference for the model to verify access control lists. This model has be extended [87] to support also IPsec gateways within the network.

Fu *et al.* [7] present a solution to verify the correct implementation of IPsec policies. The algorithm presented takes as input high-level security policies describing an implementation and compares it with a desired end-to-end policy. Even though the algorithm is able to compare a desired policy with its implementation, it cannot been used to compare a modified policy with its original version. Furthermore, it only supports IPsec policies and does not support routing or other transformation policies.

Hazelhurst [73] presents a model focused on reachability analysis and policy comparison. The model represents packet filter rules internally as BDDs and has a GUI for human interaction. The policy comparison can compare an original policy with a modified one. This model, in summary, shows if the two policies are equivalent or contain differences. The differences can be displayed in two modes, newdeny and newallow. The newdeny mode highlights the packets that are blocked by the modified policy but allowed by the original one. The newallow mode is the opposite.

Liu *et al.* [88] and [89] propose to reduce configuration errors by forcing network administrators to write two separate concrete configurations and to compare them afterwards. The two configurations are converted into two FDDs and the comparison is performed onto the two FDDs. The comparison algorithm merges the two FDDs and verifies that the action, contained in the leaves of the tree, is the same at each point. Possible anomalies found in the two FDDs must be corrected manually by the administrators and without any correlation to the original configurations. This approach can be generalized and the two input policies may be seen as the original and the modified policy.

Liu *et al.* have also published two papers on change-impact analysis of firewall policies [90, 91], their algorithm is based on a FDD and supports the classic 5-tuple filtering rules. Overlapping rules are eliminated during the creation of the FDD and as a result the FDD represents a filtering policy without overlapping rules. The algorithm is designed to support four basic operations on firewall policies: rule deletion, rule insertion, rule modification, and rule swap. The output of the algorithm presents an accurate impact of a proposed change. Furthermore, the algorithm is also capable of correlating the impact of a policy change with a high-level security requirement. Although the authors claim that the algorithm is practical, neither of the two papers does a performance evaluation of the presented algorithms.

Yin and Bhuvaneswaran [92] represent correlations between rules as spatial relations and show how this special relations can be used to evaluate the impact of rule changes on the policy. Filtering policies are represented by the so-called SIERRA tree. A SIERRA tree is similar to a FDD, each level of the tree represents a dimension of the special division. The impact analysis can only be performed on single changes, such as adding, removing or replacing rules.

The performance of the algorithm is very poor since to calculate the difference between two policies containing 30 rules takes several seconds.

Uribe *et al.* [93] propose a model for comparing a desired network security policy with configurations of a network containing packet filters and NIDS. The model verifies if the desired policy is correctly enforced.

The Margrave Tool by Nelson *et al.* [81] can perform change impact analysis of single firewall configurations. The tool can compare a modified configuration with the original one and returns the packet headers which are treated differently.

Ben Youssef *et al.* [94] propose a formal and automatic verification method based on a inference system. The solutions certify that a firewall configuration is sound and respect completely to a security policy. In case that the configuration is not sound and complete, the method provides the user with information to solve the issues. This paper only supports packet filter firewalls; however in an extended version [95], Youssef *et al.* propose a formal and automatic method to check also stateful firewall configurations.

Krombi *et al.* [49] propose a procedure that synthesizes an automaton which implements a given security policy. They use an automata-based approach to develop, in firewall policy, three analysis procedures to verify completeness, detect anomalies and functional discrepancies between several implementations of a security policy.

3.3.2 Summary of State of the Art and Research Directions

Table 3.4 shows that the major part of the considered works face the comparison between packet filter policies, while a scarce number of those comparisons consider also stateful firewall policies and IPsec policies.

As it was mentioned before, the change impact analysis and implementation verification can be carried out during a policy comparison. Although the implementation of both aspects could make the policy comparison more complete, none seems to use both in the same work. Even if the whole literature presents

a prototype of the proposed solution, only few of those policy comparison prototypes have been validated into real use cases.

In summary, we think that the research is only at the beginning and much more improvements can be introduced there.

	Data			Policy Type			Features		Validation			
	Y	AUT	CIT	PKF	IPsec	APF	CHN	IMPL	PRT	TST	EMP	THEO
[86]	1997	Guttman	63	✓				✓	✓	✓	✓	
[87]	2005	Guttman	30	✓	✓			✓	✓			
[7]	2001	Fu	29			✓		✓	✓			
[73]	2003	Hazelhurst	60	✓				✓	✓			
[88]	2004	Liu	29	✓			✓		✓	✓	✓	
[90]	2007	Liu	6	✓			✓					
[89]	2008	Liu	66	✓			✓		✓	✓	✓	
[91]	2012	Liu	5	✓			✓		✓	✓	✓	
[92]	2006	Yin	4	✓			✓		✓	✓		
[93]	2007	Uribe	27	✓				✓		✓		
[94]	2009	Ben Youssef	5	✓			✓		✓	✓		
[95]	2011	Ben Youssef	4			✓		✓	✓			
[49]	2014	Krombi	2	✓				✓				

Y= Year; AUT= Author (reference author); CIT= Citation numbers (scopus and/or scholar); PKF=Packet Filter; SFF= Stateful Firewall; CHN= Change Impact Analysis; IMPL= Implementation Analysis; PRT= Prototype; TST= Test; EMP= Empirical evaluation; THE= Theoretical validation

Table 3.4 Summary of state of the art in Policy Comparison

Chapter 4

Network Security Policy: Contributions

Anomalies in the network security configurations may result in serious breaches and vulnerability causing problems such as blocking legitimate traffic, permitting unwanted traffic and sending insecure data. Therefore we need a *Policy Anomaly Analysis*, in order to identify potential errors, conflict and redundancy among policy rules.

Although the literature has proposed several works and techniques to identify anomalies, it is mainly focused on Intra- and Inter-policy analysis. The *Intra-Policy* analysis identifies any anomaly in the rules of a single policy, while the *Inter-Policy* analysis identifies anomalies in rules of a set of interconnected policies.

Those works have been carried on in different policy application domains: communication protection, filtering, traffic flow policy and others. Unfortunately, such works have not been applied into the real activities of a network administrator, as well as the major part of them generally is limited to a specific policy domain.

For this reason, our contribution to the state-of-the-art is the definition of innovative anomaly analysis for network security policy based on two new classes of policy anomaly: Inter-technology and Inter-domain.

The *Inter-technology Policy Analysis* identifies any anomaly in a set of policies of different security communication technologies (e.g. IPsec, TLS, SSH). For instance, when an IPsec tunnel encapsulates other TLS tunnels, the external tunnel is a redundant level of protection.

The *Inter-domain Policy Analysis* identifies anomalies among policies of different domains. This is the case of a firewall that blocks some encrypted communication channels created by a VPN functionality, which generates an Inter-domain anomaly between the filtering domain (i.e. the firewall) and the communication-protection one (i.e. the VPN).

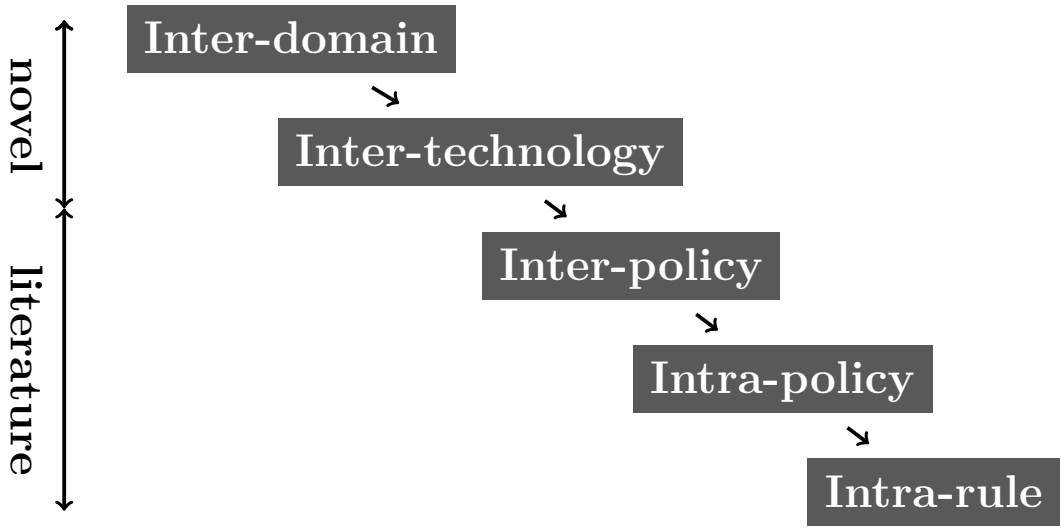


Fig. 4.1 Policy Analysis Classes.

We divide the rest of this thesis in two parts: the first part focuses on the communication protection policy analysis, where we develop a model of communication protection policy, used in an Inter-technology analysis where many network security technologies have been considered; while the second part presents a Unified Model for Policy Analysis (UMPA) that embraces the analysis model applied into single policy domain into a comprehensive analysis of anomalies among many policy domains (i.e. Inter-domain anomaly analysis).

4.1 Inter-technology analysis of communication protection policies

In this part of the thesis, we pushed forward the state-of-the-art in several directions.

The main contribution is the identification of nineteen types of anomalies that may happen when implementing a Communication Protection Policy (CPP). Six anomaly types are already known [57], but all the others are our original contribution.

Anomalies are detected by means of a formal model that takes as input the network description, nodes information and, the security controls' configurations and the communications to secure (during a CPP design validation) or the communications actually secured (during a CPP implementation analysis). Information in input become part of a knowledge base explored with a set of FOL formulas to identify and report the detected anomalies to the administrators.

Anomalies have been categorized according to two different classifications: an *effect-based* taxonomy and an *information-centric* one. The first classification divides the anomalies describing the effects that they have on the network, while the second one is based on the information needed to be analysed for detecting the anomalies.

Having introduced several novel kinds of anomalies, we posed ourselves some questions regarding the impact of our work. First of all, we have to evaluate the effects of the detection of such new anomalies in improving the security of the current IT infrastructures. We have to also understand if the newly-introduced anomalies may be really generated by the administrators during the configuration phase and, in case it is, we have to evaluate the costs to detect such anomalies in large networks.

In order to answer the first question, for each anomaly we present the possible consequences on the network and some ways to resolve it for reducing the security impacts on the short and long period.

To answer the second question, we have prepared an empirical experiment where three categories of administrators (experts, intermediate and beginners)

were asked to configure a set of CPPs in a sample network. We show how several of the newly introduced anomalies appear.

And finally, to answer the last question, we implemented, and tested in several different scenarios, a tool making use of DL (description logic) ontologies and custom Java-based reasoning rules.

4.2 Definition of Unified Model for Policy Analysis

Our contribution is to define a unified formal model for detecting network conflicts and irregularities by defining *Inter-* and *Intra-domain* policy anomalies, in order to avoid erroneous and unexpected network behaviours.

In particular, we design a model that is general enough to collect under its umbrella different types of network policies, and exploits a medium-level policy representation, named *Policy Implementation* (*PI*).

A medium-level representation avoids network administrators to use vendor-specific and implementation-dependent interfaces, required by a given security control.

Let us consider the case of two firewalls provided by the same vendor that support different configuration methods, for example the SNMP protocol for the first firewall and the command line interface (CLI) for the latter. In this case, an administrator should detect configuration errors and anomalies on his firewalls by checking manually functions configurations. This implies that the administrator has a deep knowledge about the two firewalls, the supported methods of configuration and the ways to get the installed configuration in order to check it and, in case of errors, fix it.

Here, an automatic translation of the low-level configuration of a network function into a set of PIs may improve the anomaly analysis making it implementation-independent and easier. However, this thesis is mostly related to the definition of the analysis approach, thus we leave this translation as future work.

In addition, this medium-level policy analysis can be added to a *process of policy refinement*. In this way, we can reduce the anomalies deriving from the refinement process itself, and produce final configurations that are anomaly-free.

In summary, the PI representation facilitates the analysis of network and security configuration through a top-down approach, starting from high-level policies to low-level anomaly-free configurations. In addition, it could also enable an anomaly verification process that uses a bottom-up approach by translating the low-level configurations into PIs.

The PI-based analysis model is composed by three parts: *(i)* the definition of a hierarchical structure for the PIs to identify the relations between network policies; *(ii)* an anomaly classification to specify which set of policy conditions imply an anomaly; *(iii)* anomaly detection rules to identify when a certain network policy anomaly occurs. Those components are described in detail in this part of the thesis.

Finally in order to validate the generality of this solution, we considered three case studies to be mapped into the model. We have selected a case-study already known in the literature, that is the filtering policy, a new example of security policies like the communication protection one and finally a novel case of non-security-related policies, which are the traffic flow policies.

Communication Protection Policies

Chapter 5

Communication Protection Policies: Modelling and Analysis

As defined in Chapter 2, Communication Protection Policies (CPPs) allow the definition of how to make the network communications secure in critical contexts, like financial or corporate.

Several proposals have been presented in literature for detecting CPP conflicts, but most of them focus only on a single security technology (i.e. IPSec) and overlook the effects of multiple overlapping protection techniques (see Chapter 3).

In this chapter, we present a novel classification of communication protection policy anomalies and a formal model which is able to detect anomalies. The result of our analysis allows administrators to have a precise insight on the various alternative implementations, their relations and the possibility of resolving anomalies, thus increasing the overall security and performance of a network.

5.1 Background

In this section, we briefly introduce the main concepts and terms that we will use throughout the rest of the work.

A *communication* is any directional data exchange between two network entities. A *secure communication* is a communication 'adequately' protected, that is, it fully satisfies a set of security requirements. In this context, the security requirements concern three *security properties*: header integrity, payload integrity and (payload) confidentiality.

A *channel* is directional data exchange between two nodes protected with some security properties (a *secure* channel) or none (an *insecure* channel). Logically, a secure channel is an association between a source node, where the security properties are applied, and a destination node where the security properties are removed or verified. A communication can be thought as a stack of several (secure and/or insecure) channels. For example, an end-to-end TLS communication consists of a single secure channel. More complex scenarios exist, however. For instance, a communication between two hosts in two separated networks connected with an IPsec site-to-site VPN is modelled with two channels: an insecure one between the end-points and an IPsec secure channel between the VPN terminators.

In the real world, the secure communications are defined by using a set of configuration settings containing several low-level details. For instance, the configuration of a TLS server contains detailed information about the supported cipher-suites. During the design and policy analysis phases, however, this level of granularity is usually not needed. For our purposes, a secure channel can be represented by specifying:

1. the source and destination entities (they can be network nodes or direct references to an entity lying at a particular OSI layer such IP addresses and URIs);
2. the security protocol to use (our model can be easily extended to new protocols and can support a wide array of technologies at different OSI layers);
3. the required security properties;
4. the crossed gateways and the traffic to protect (meaningful only in case of tunnels).

We name *policy implementation* (or *PI* for short) this formal representation of a channel. Note that since a channel is directional, a PI is directional too. This means that, to create a complete request-reply connection, we need at least two PIs. We call a *PI set* a group of policy implementations that belongs to the same node and use the same technology. For instance, a particular server supporting IPsec and SSH will have two PI sets, one for each protocol. We will assume without loss of generality that the policy implementations in the same PI set are ordered according to their priority

Note that to perform our analysis, we will make use of other additional sources of information such as:

- *network reachability data*, as the configurations of filtering controls and NAT devices must be available to determine if the channels can be actually established (e.g. to check if a channel is not dropped by a firewall);
- *supported security protocols* (at various OSI levels), for guaranteeing that it is possible to establish the secure channels;
- *supported cryptographic algorithms*, as some cipher-suites might not be available when actually deploying a PI on the installed security controls.

Finally, we will refer to the *network topology* as a graph where its nodes are potential channel end-points (both sources and destinations) and its edges are physical or virtual connections between them.

5.2 Motivating example

Before formally tackling the analysis of the PI anomalies, we will begin our discussion by considering the simplified network scenario presented in Fig. 5.1.

The diagram shows a main corporate network (C) and two branch networks (A and B). The three networks are connected through the Internet and consist of a number of security gateways (denoted by the letter g) that mediate the communications between servers (s_{c1} and s_{c2}) and clients (indicated by the symbol c). Server s_{c1} hosts two services (web_1 and db), while s_{c2} hosts only one web service (web_2).

From now on, we use the informal notation $s \xrightarrow[pi,c]{\text{t}} d$ to indicate a PI that establishes a channel from the source s to the destination d using the technology t to enforce some security properties. In this simplified example we will only take into account two security properties, (payload) confidentiality and payload integrity, denoted by the symbols c and pi, respectively. For instance, the PI $a \xrightarrow[pi]{\text{IPsec}} b$ indicates an IPsec connection with integrity (but not confidentiality), from a to b.

For the sake of clarity, in this example we present the communication protection policy anomaly using the *effect-based* taxonomy. As shown in Fig. 5.2, this classification divides the anomalies into five macro-categories: 1) insecure communications; 2) unfeasible communications; 3) potential errors; 4) suboptimal communications; 5) suboptimal walks.

These macro-categories will be briefly described in the following paragraphs.

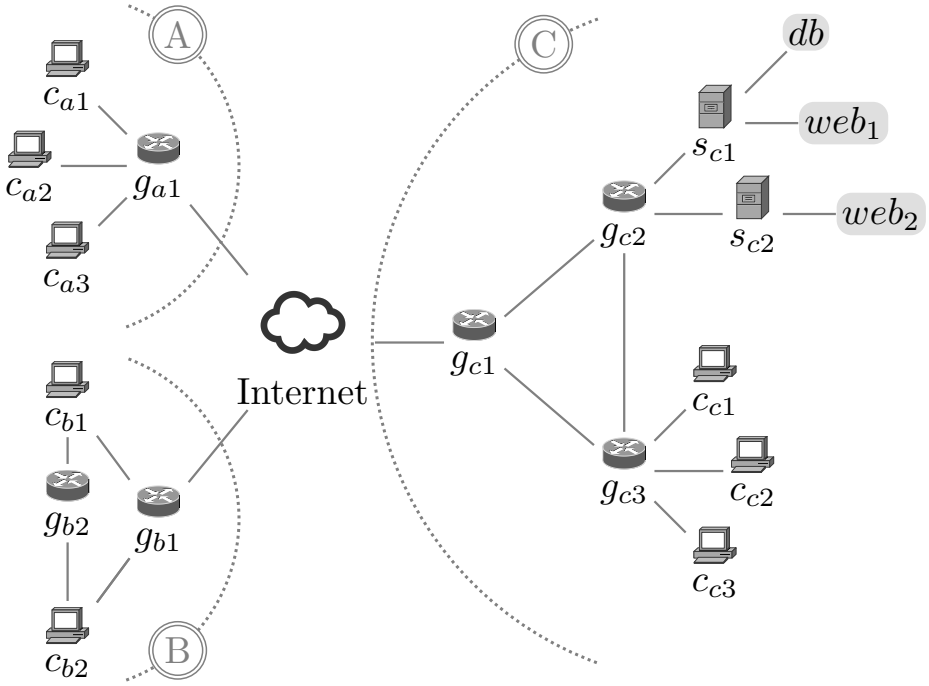


Fig. 5.1 A simplified network scenario.

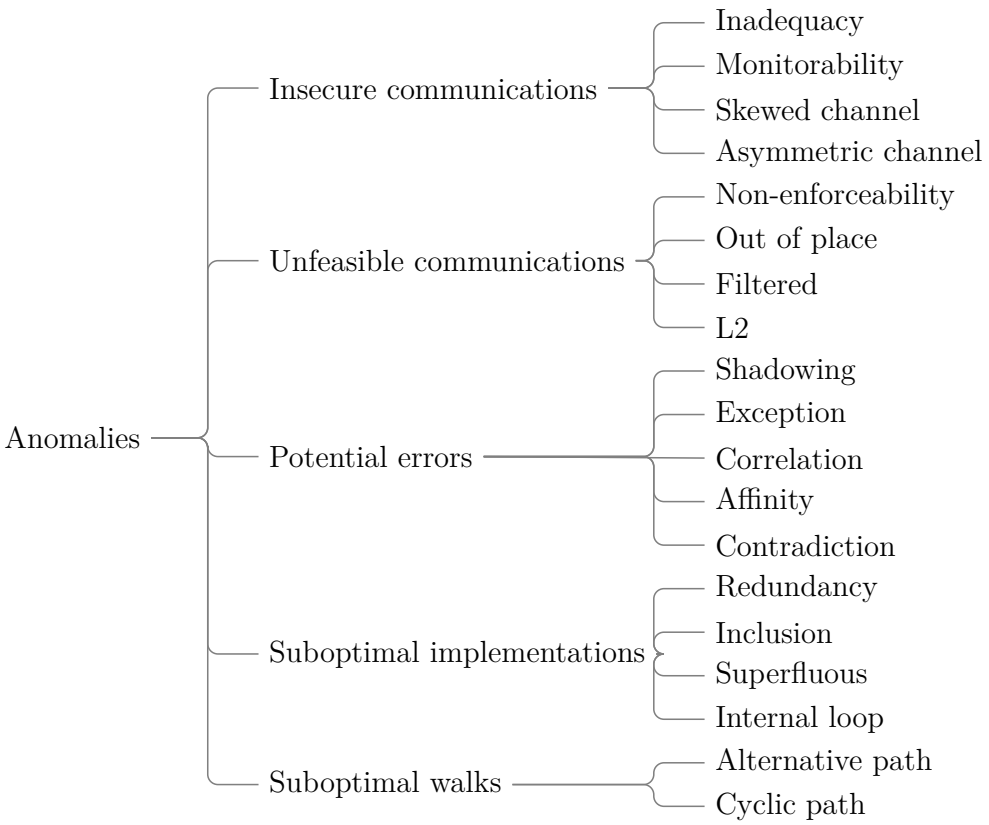


Fig. 5.2 Effect-based taxonomy of CPPs.

5.2.1 Insecure communications

An insecure communication occurs when communication security level is lower than the expected one. For instance, we can have a channel that does not satisfy the minimum security level, e.g. specified in the corporate policy, thus creating an *inadequacy* anomaly. For example, we may observe such anomaly when the IT managers require that 'all the data crossing Internet must be encrypted' and a security administrator creates the policy implementation $c_{a1} \xrightarrow[\{pi\}]{\text{TLS}} s_{c1}$.

Another case of insecure communication arises when the security requirements are respected but the communications consist of more than one channel (e.g. remote access). Here, the nodes at the channel junctions can 'see' the exchanged data, thus lowering connection security and creating a *monitorability* anomaly (Fig. 5.3). For instance, the PIs $s_{c1} \xrightarrow[\{c,pi\}]{\text{IPsec}} g_{c1}$ and $g_{c1} \xrightarrow[\{c,pi\}]{\text{IPsec}} c_{a1}$ create a sort of logical communication between s_{c1} and c_{a1} composed of a sequence of

two channels interconnected through g_{c1}. This means that, even if everything is encrypted, g_{c1} reads the payload since it decrypts and encrypts again the exchanged data.

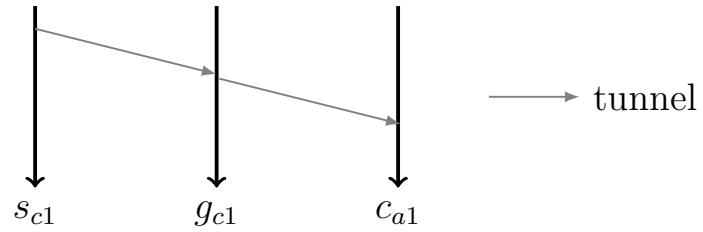


Fig. 5.3 Diagram of the monitorability between s_{c1}, g_{c1}, c_{a1}.

Another kind of insecure communication, - more subtle but potentially catastrophic- can occur with a wrong tunnel overlapping that removes the confidentiality in a part of the communication and produces a *skewed channel* anomaly (a super-set of Al-Shaer's overlapping anomalies [57]). For example, a security administrator can create a tunnel $g_{c3} \xrightarrow[\{c\}]{\text{IPsec}} g_{a1}$ and another one with $g_{c3} \xrightarrow[\{c\}]{\text{IPsec}} g_{c1}$ (note that the latter tunnel is 'included' in the first one). The trellis diagram in Fig. 5.4 helps to graphically visualize the problem.

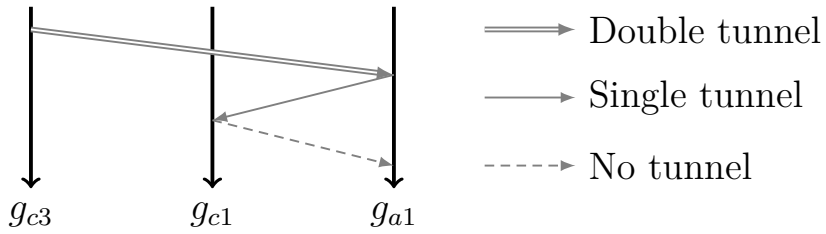


Fig. 5.4 Diagram of the skewed channel between g_{c3}, g_{c1}, g_{a1}.

When g_{c3} sends some data, it encapsulates the information in two tunnels. Hence, when g_{a1} receives the data, it removes the external tunnel encapsulation without removing, the internal one. Then, g_{a1} sends the data back to g_{c1} which, in turn, removes the last tunnel. Finally, g_{c1} sends the data to g_{a1} with no protection, thus exposing the communication content to a sniffing attacker.

In the real world, the majority of connections are bidirectional, since a request usually requires a reply. It may occur that the request channel has a different security level from the reply one, generating an *asymmetric channel* anomaly. This is not necessarily an issue, but it could be useful to report this

inconsistency to the administrators, so that they can check if security control configurations reflect the intended network behaviour.

5.2.2 Unfeasible communications

An *unfeasible communication* is a communication that cannot be established because of a hard misconfiguration. These anomalies are very severe since they completely prevent any data exchange.

The simplest example of an unfeasible communication is when security administrators design a PI with a technology that is not supported by an endpoint or with a security level that is too high to be enforced by the available cipher-suites. This situation is defined as a *non-enforceability* anomaly. For instance, the policy implementation $web_2 \xrightarrow[\{c\}]{\text{TLS}} db$ becomes non-enforceable if the service administrators did not install TLS on s_{c2} (where web_2 resides). This PI must obviously be deployed on the source endpoint s_{c2}. However, if the node containing this PI is not s_{c2}, we have generated a further issue: an *out of place* anomaly.

We also have a hindered connection if the packets of a channel are dropped by a firewall that lies on the path between source and destination, thus producing a *filtered channel* anomaly.

Firewalls and bad server configurations are not the only causes of an unfeasible communication. There are also technological incompatibilities between wired and wireless protocols when performing security at level 2 (data link) of the ISO/OSI stack. For example, if we choose to create a secure channel using WPA2 technology, we must be sure that the network frames only cross wireless-enabled nodes. If one or more crossed nodes are wired-only, then we have an *L2 anomaly*.

5.2.3 Potential errors

Potential errors form a class of anomalies where the original intent of the administrators is unclear. Hence their solution requires a thorough human inspection.

When working with a large group of PIs, an administrator can create a PI that meddles all the traffic of another PI with different security properties. For instance, the PI $c_{a1} \xrightarrow[\{pi\}]{\text{TLS}} web_1$ hides $c_{a1} \xrightarrow[\{c\}]{\text{TLS}} web_1$, if the first one has a higher priority. Since the former shadows the latter, such an anomaly is called *shadowing* anomaly, while when the second PI has a higher priority, we have an *exception* anomaly. Exceptions are useful and are typically exploited by administrators to express an 'all but one' rule, even though we report them for verification.

Another kind of potential errors occurs when there are two PIs sharing the same technology and the same source and destination on the same node. This situation can lead to an ambiguity, since sometimes a piece of data can match multiple PIs, hence making the intended protection level unclear. For example, $web_2 \xrightarrow[\{c\}]{\text{TLS}} s_{c1}$ and $s_{c2} \xrightarrow[\{c,pi\}]{\text{TLS}} db$ are ambiguous, for a packet from web_2 to db can match both PIs. This problem is a *correlation* anomaly. Similarly, there will be an *affinity* anomaly between two PIs using different technologies but having source and destination on the same nodes.

Finally, we have a *contradiction* anomaly when two PIs respectively express that the same communication should be protected and unprotected. For instance, let suppose that an IT manager defines a policy where 'all the traffic for the Internet must be inspected' and a security manager enforces the encryption of the traffic exchanged between c_{a1} and s_{c1} via the PI $c_{a1} \xrightarrow[\{c,pi\}]{\text{IPsec}} s_{c1}$. This leads to a contradiction, since the policy requires that the data for the Internet should be monitorable, while PI encrypts them.

5.2.4 Suboptimal implementations

Suboptimal implementations arise when one or more PIs can decrease the network throughput by producing some overhead in the nodes. Their existence is not usually problematic, but their solution can be beneficial, since it improves network performance and makes the PIs less vulnerable to DoS attacks.

The simplest kind of suboptimal implementations occurs when an administrator deploys a PI that makes another PI useless, as the first one can secure the communication at the same, or at a higher level than the second one, with a more effective protection (e.g. stronger encryption). For example, two different

security administrators may have independently defined the PIs $c_{a1} \xrightarrow[\{c\}]{\text{TLS}} web_1$ and $c_{a1} \xrightarrow[\{c,pi\}]{\text{IPsec}} s_{c1}$. The former is included in the latter, so that it can be safely removed. In these cases, we may observe a *redundancy* anomaly if both the PIs use the same technology, or an *inclusion* anomaly if they use two different protocols.

Another type of suboptimality arises when a tunnel encapsulates other tunnels with a higher security level. This is a *superfluous* anomaly and can be resolved by simply deleting the external, redundant tunnel.

There may also be some channels that can be safely removed without altering the network semantic. This happens in the so called *internal loop* anomalies, where a PI source and destination belong to the same node.

5.2.5 Suboptimal walks

A group of PIs can produce a *suboptimal walk* when the path taken by the data is unnecessarily long.

In large networks, a communication between two end-points can take multiple paths, thus generating an *alternative path* anomaly. They are not necessarily a misconfiguration, but we can detect and report them to the administrators as a safety measure. For example, $g_{c2} \xrightarrow[\{c\}]{\text{IPsec}} g_{c3}$ forms an alternative path w.r.t. $g_{c2} \xrightarrow[\{c\}]{\text{IPsec}} g_{c1}$ and $g_{c1} \xrightarrow[\{c\}]{\text{IPsec}} g_{c3}$.

Another cause of suboptimality occurs when some data cross a node multiple times during their travel. This *cyclic path* anomaly can be removed by deleting the cycles, thus shortening the network path.

5.3 PI hierarchical structure

In this section, we will formally define what a policy implementation is, as well as defining its structure and the relationships between the various network fields that compose it. In addition, we will also describe the notion of path, which is used to detect several kind of network anomalies.

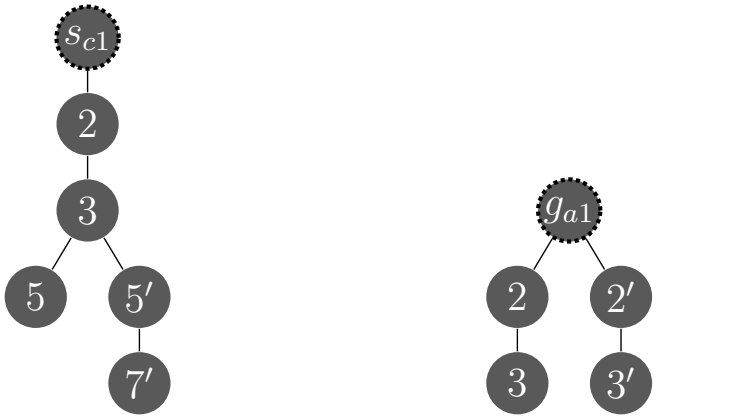
In our model, a PI i is a tuple:

$$i = (s, d, t, C, S, G)$$

Where:

- s and d respectively represent the channel source and the destination (Section 5.3.1);
- t is the security technology adopted (Section 5.3.2);
- C is an ordered set of coefficients that indicate the security levels required (Section 5.3.3);
- S is a selector, i.e. a tuple of network fields, used to identify the traffic that needs to be protected (Section 5.3.4);
- G is the list of the gateways involved in the communication (Section 5.3.5).

5.3.1 Sources (s) and destinations (d)



(a) Representation of s_{c1}. (b) Representation of g_{a1}.

Fig. 5.5 Graphical representation of a server and a gateway.

To perform an accurate detection of anomalies, it is necessary to identify the layer in the OSI stack where a secure communication starts and terminates. To this purpose, we use a hierarchical structure that represents the points

where the secure communication end-points can be established. This structure has an elementary tree-like graphical representation as shown in Fig. 5.5.

The root represents the network node itself, while all the tree nodes model the available *network entities* in the OSI stack. In this paper we will only focus our attention on the data link, network, session and application layers. The tree levels may also be associated respectively to the layer 2 addresses, IP addresses, port numbers and URIs. To avoid ambiguity, we will use the notation $s_{c1}.l5'$ to specify the node labeled $5'$ in the s_{c1} tree and so on.

Note that the gateways expose multiple interfaces, one for each network where they are connected to. For instance, in Fig. 5.5b, we have two vertices at layer 3 that represent the 'internal' interface for the network A and an 'external' interface for the Internet. If a gateway supports also VPNs creation via TLS tunnels (e.g. OpenVPN), we will have two additional vertexes at the session level.

Given any two network entities e_1 and e_2, we define the following relationships:

- e_1 is *equivalent* to e_2 ($e_1 = e_2$) if they are exactly the same entity;
- e_1 *dominates* e_2 ($e_1 \succ e_2$) if all the traffic starting from (or arriving to) e_2 passes through e_1. Graphically, that means that e_1 is an ancestor of e_2 in the tree representation. This concept is particularly useful when dealing with security protocols working at different OSI layers. In Fig. 5.5a, for instance $s_{c1}.l3$ dominates $s_{c1}.l7'$.
- e_1 is a *kin* of e_2 ($e_1 \sim e_2$) if e_1 and e_2 belong to the same network node, but there is no equivalence or dominance relationship amongst them. For example, in Fig. 5.5a, $s_{c1}.l5$ is a kin of $s_{c1}.l5'$.
- e_1 and e_2 are *disjoint* ($e_1 \perp e_2$) if they belong to different network nodes (and hence trees).

Note that if e_1 and e_2 are not disjoint ($e_1 \not\perp e_2$) , it means that they are on the same device. Hence, they are related by an equivalence, dominance or kinship relationship.

5.3.2 Technologies (t)

In this thesis, we take into account a limited set of technologies. Nonetheless, our model is flexible enough to accommodate any security protocol. In particular, we will only consider:

- the data link layer: WPA2 and 802.1AE MACsec;
- the network layer: IPsec;
- the session layer: TLS[1];
- the application layer: WS-Security and SSH.

In addition, we also use the special NULL technology, to indicate that a communication should be created without any kind of protection.

Similarly to network entities, two technologies t_1 and t_2 can have different relationships:

- t_1 is *equivalent* to t_2 ($t_1 = t_2$), if they are exactly the same technology;
- t_1 *dominates* t_2 ($t_1 \succ t_2$) if the OSI layer of t_1 is strictly lower than the t_2's one. The NULL technology is dominated by all the other technologies.
- t_1 is a *kin* of t_2 ($t_1 \sim t_2$) if t_1 and t_2 are different and they work at the same OSI layer;
- t_1 is *disjoint* from t_2 ($t_1 \perp t_2$) if one technology is NULL and the other one is not NULL.

In general, the following relationships hold:

$$t^{(i)} \sim t'^{(i)}, \quad t^{(i)} \neq t'^{(i)}$$
$$t^{(2)} \succ t^{(3)} \succ t^{(5)} \succ t^{(7)}$$
$$t \perp \text{NULL}, \quad \forall t \neq \text{NULL}$$

[1]Protections at transport layer, such as TLS, are sometimes associated to the session layer as they work on top of the TCP/UDP protocols. We do not want to enter a philosophical diatribe as, for our purposes, the important thing is the order of encapsulation of the different protections.

Where $t^{(i)}$ represents a technology at the OSI level i.

5.3.3 Security coefficients (C)

The set of security coefficients consists of several non-negative real values that indicate a required security level for a specific property. The higher a value, the stronger the enforcement of a property should be. On the other hand, if a coefficient is zero the related security property must not be enforced. Obviously, if the chosen technology is NULL, all the coefficients are zero. These values should be estimated by the administrators with the use of some metrics, for example on the chosen cipher-suite (taking into account the key length, encryption/hash algorithms and cipher mode).

In this thesis, we only focus our attention on three properties, which are header integrity (c^{hi}), payload integrity (c^{pi}) and (payload) confidentiality (c^{c}), so that:

$$C = (c^{hi}, c^{pi}, c^{c})$$

The relationships amongst two coefficient sets C_1 and C_2 are:

- C_1 is *equivalent* to C_2 ($C_1 = C_2$) if all the coefficients of C_1 are the same as their C_2's counterparts;
- C_1 *dominates* C_2 ($C_1 \succ C_2$) if at lest one coefficient of C_1 is strictly greater than its C_2's counterparts and the other coefficients of C_1 are not lower than their C_2's counterparts;
- C_1 is *disjoint* with C_2 ($C_1 \perp C_2$) if there is neither dominance nor equivalence between C_1 and C_2, that is $C_1 \not\succeq C_2 \wedge C_1 \not\preceq C_2$.

5.3.4 Selectors (S)

As presented in Chapter 2, some security protocols (e.g. IPsec, see RFC-3585 [96]) allow the definition of filtering conditions to select the traffic that must be protected. Our model supports such conditions via the *selectors* S of a policy implementation, that are tuples of network fields. In theory (and in our model too), S can be arbitrarily defined with any field, however, in practice,

the fields in S are usually the well-known five-tuple consisting of a source IP address (ip_{src}) and port (p_{src}), a destination IP address (ip_{dst}) and port (p_{dst}) and a protocol type (prt). We will assume this in the rest of the paper that S at least includes the five-tuple, that is:

$$S = (ip_{src}, ip_{dst}, p_{src}, p_{dst}, prt, \ldots)$$

We will use the notation $\overleftarrow{S}$ to indicate a *reverse* list of selectors where source and destination are swapped, that is $\overleftarrow{S} = (ip_{dst}, ip_{src}, p_{dst}, p_{src}, prt, \ldots)$.

In addition, we use the notation $S|_{f_1 \times f_2 \times \ldots}$ to restrict the selector space to the fields f_1, f_2, ... For instance, in the following sections, we will often use the more compact $S|_{ip_{src} \times ip_{dst}} = (ip_{src}, ip_{dst})$ instead of the n-tuple $(ip_{src}, p_{dst}, *, *, *, \ldots)$ (where the asterisk symbol $*$ will denote a field matched by any value) to define a selector that matches all the traffic from ip_{src} to ip_{dst} regardless of the port numbers and protocol. Moreover, the all-matching tuple $(*, *, *, *, *, \ldots)$ will be also shortened to a single $*$ inside a PI definition.

In addition, we will make use of the following relationships between the selector tuples:

- S_1 is *equivalent* to S_2 ($S_1 = S_2$), if their selectors are exactly the same;
- S_1 *dominates* S_2 ($S_1 \succ S_2$) if the matched traffic of S_2 is a sub-set of the matched traffic of S_1;
- S_1 is a *kin* to S_2 ($S_1 \sim S_2$) if there is at least one communication that matches S_1 but not S_2 and vice-versa;
- S_1 is *disjoint* from S_2 ($S_1 \perp S_2$) if the sets of the traffic matched by S_1 and S_2 are disjoint.

5.3.5 Crossed gateways (G)

Tunnel PIs contain an ordered set G that specifies the gateways crossed by the channel traffic. The G set of tunnel PIs is statically computed from the network topology and the content of the routing tables.

Note that the list of crossed gateways does not contain the channel source and destination nodes. We use the notation G^* to indicate a list containing also the PI end-points, that is $G^* = \{s\} \cup G \cup \{d\}$. We will also denote the list of crossed gateways in reverse order with $\overleftarrow{G}$.

It is worth presenting an example of PIs that use gateways. Given the network in Fig. 5.1, a communication from c_{a1} to s_{c1} that passes into an IPsec tunnel between the two gateways g_{a1} and g_{c2} is implemented by two PIs:

$$i_1 = (c_{a1}, s_{c1}, \text{NULL}, (0,0,0), *, (g_{a1}, g_{c1}, g_{c2}))$$
$$i_2 = (g_{a1}, g_{c2}, \text{IPsec}, (3,3,3), ip_{c_{a1}}, *, ip_{s_{c1}}, *, *), (g_{c1}))$$

where i_1 specifies the communication that will be encapsulated in the tunnel defined by i_2.

5.3.6 Paths

We introduce now the concept of path, which is strictly related to the notion of policy implementation. The notation P^{e_1,e_n} represents a possible path starting from the network entity e_1 and terminating into the network entity e_n. Each *path* is a list of policy implementations $(i_1, i_2, \ldots, i_n)$ where: (i) the source of the first PI i_1 is e_1; (ii) the destination of the last PI i_n is e_n; (iii) given two consecutive PIs in the path i_j and i_{j+1}, the property $d_j \in S_{j+1}$ holds.

For instance, a path from c_{c2} to s_{c2} can be implemented by:

$$i_1 = (c_{c2}, g_{c3}, \text{NULL}, (0,0,0), *, \varnothing)$$
$$i_2 = (g_{c3}, g_{c2}, \text{IPsec}, (3,3,3), (subnet_{c_c}, *, subnet_{c_s}, *, *), \varnothing)$$
$$i_3 = (g_{c2}, s_{c2}, \text{NULL}, (0,0,0), *, \varnothing)$$

Since P^{e_1,e_n} is a set, we will use the notation $|P^{e_1,e_2}|$ to indicate its cardinality, that is the number of policy implementations that compose it.

Note that two paths $P_1^{e_1,e_n}$ and $P_2^{e_1,e_n}$ are different ($P_1^{e_1,e_n} \neq P_2^{e_1,e_n}$) if they differ by at least one element and/or if their respective PIs are placed in different orders.

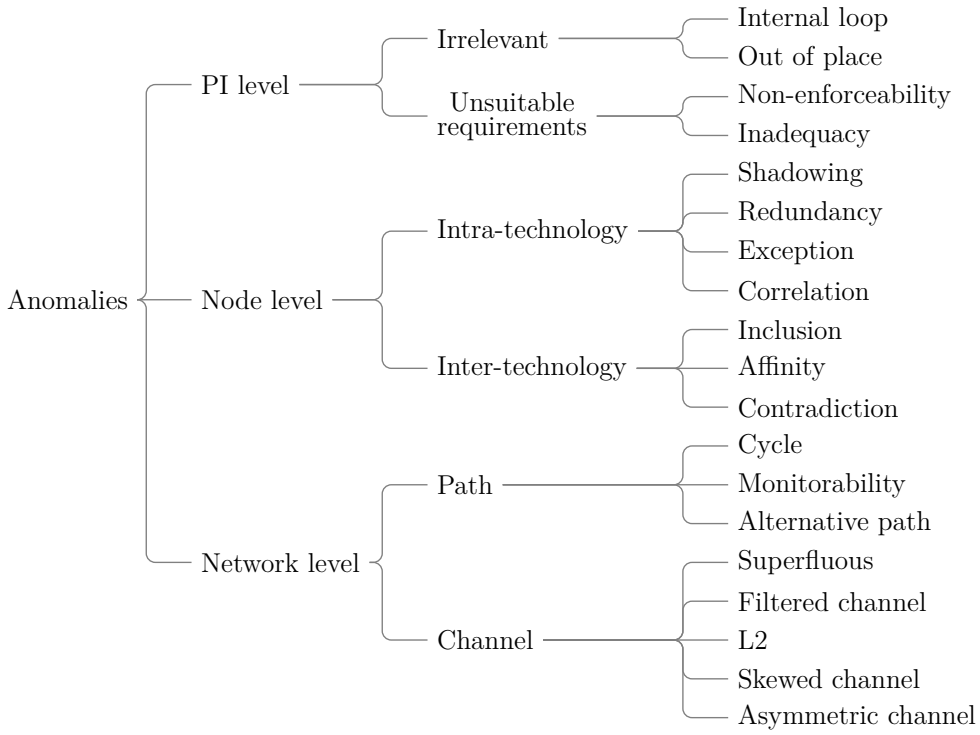


Fig. 5.6 Information-centric taxonomy of CPPs.

5.4 Anomaly analysis and resolution

Having formalized the definition of a policy implementation, we express the logic formulas used to detect the various anomalies.

In Section 5, we introduced an anomaly classification - which emphasizes the side effects of an anomaly- based on five macro-categories (Fig. 5.2). In the following paragraphs, however, we will adopt a more technical classification (Fig. 5.6), which suits better for a formal discussion. As a matter of fact, such a classification highlights the possible levels of interactions among PIs and, hence, at which level it is possible to generate an anomaly.

We distinguish three levels of anomalies: 1) the *PI level anomalies* that occur within a single PI; 2) the *node level anomalies*, which come up between two distinct PIs placed on the same node; 3) the *network level anomalies* arising between distinct PIs that belong to different nodes.

5.4.1 PI level anomalies

We can distinguish two families of PI level anomalies: irrelevant and unsuitable requirement anomalies. A PI that generates an *irrelevant anomaly* (internal loop and out of place) is meaningless for the network semantics, so that their presence does not change how network exchange data. The *unsuitable requirement anomalies* (non-enforceability and inadequacy), instead, break some security requisite and can lead to severe problems.

Internal loop – $\mathcal{A}_{il}(i_1)$

An *internal loop anomaly* occurs when source and destination end-points are on the same node, thus creating a data loop. These anomalies can be inferred using the formula:

$$\mathcal{A}_{il}(i_1) \Leftrightarrow s_1 \not\perp d_1$$

The resolution method proposed is to simply delete i_1.

Out of place – $\mathcal{A}_{op}(i_1)$

An *out of place anomaly* consists of a PI deployment on a wrong network node, which means that the source is disjoint with the node where the PI is deployed. In order to detect these anomalies, we use the function $\mathcal{N}(i_1)$ that returns the node where the PI is actually deployed. The formula is then:

$$\mathcal{A}_{op}(i_1) \Leftrightarrow \mathcal{N}(i_1) \perp s_1$$

The simplest resolution is to delete i_1. However, a more suitable approach can be to redeploy the PI on the correct node or to appropriately modify its source.

Non-enforceability – $\mathcal{A}_{ne}(i_1)$

A PI i_1 is non-enforceable when its technology is not supported by the source, the destination or when its security coefficients are 'too high', and hence cannot be enforced.

We will make use of two functions: $\mathcal{T}(e)$, which returns the set of technologies supported by the node e, and $C_{max}(i_1)$, which returns the set of maximum enforceable coefficients by the PI i_1.

These anomalies can be identified with the formula:

$$\mathcal{A}_{ne}(i_1) \Leftrightarrow C_1 \succ \mathcal{C}_{max}(i_1) \vee t_1 \notin \mathcal{T}(s_1) \vee t_1 \notin \mathcal{T}(d_1)$$

To solve such anomalies an administrator can choose to upgrade the security libraries/services on the PI source/destination to support the desired technologies or, alternatively, he might modify the PI by changing the protocol or lowering the security coefficients.

Inadequacy – $\mathcal{A}_{in}(i_1)$

We have an *inadequacy anomaly* when the security coefficients of a policy implementation establish a channel lower than an acceptable threshold. We can use a function $C_{min}(i_1)$ that returns the minimum acceptable coefficients for the channel defined by the PI i_1. This function should be defined a priori by the administrators according to some sort of metric or best practice [97–99]. For example, a network administrator could define a function such as:

$$C_{min}(i_1) = \begin{cases} \langle 1,1,1 \rangle & \text{if } i_1 \text{ is crossing the Internet} \\ \langle 0,0,0 \rangle & \text{otherwise} \end{cases}$$

We can detect these anomalies with the rule:

$$\mathcal{A}_{in}(i_1) \Leftrightarrow C_1 \prec \mathcal{C}_{min}(i_1)$$

In order to fix these issues, security requirements of policy implementation must be increased so that the property $C_1 \succeq \mathcal{C}_{min}(i_1)$ holds.

5.4.2 Node level anomalies

A *node level anomaly* occurs between two distinct policy implementations lying on the same node.

If the two PIs share the same technology then we have an *Intra-technology anomaly* (shadowing, exception, redundancy and correlation anomalies). Otherwise, there will be an *Inter-technology anomaly* (inclusion, affinity and contradiction anomalies). The Intra-technology anomaly category has been heavily inspired by the work of Al-shaer *et al.* [57].

For detecting these anomalies, we assume that the two PIs have the same crossed gateways, i.e. $G_1 = G_2$. In addition, we will also make use of the function $\pi(i) \in \mathbb{N}$ that returns the priority of a PI in a PI set (the lower the number the higher the priority).

Shadowing – $\mathcal{A}_{sh}(i_1, i_2)$

A PI i_2 is *shadowed* when there is another policy implementation i_1 with a higher priority that matches all the traffic of the first one ($s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge S_1 \succeq S_2$) and has disjoint security coefficients. We can detect these anomalies using the formula:

$$\mathcal{A}_{sh}(i_1, i_2) \Leftrightarrow \pi(i_1) < \pi(i_2) \wedge t_1 = t_2 \wedge s_1 \succeq s_2 \wedge$$
$$\wedge d_1 \succeq d_2 \wedge S_1 \succeq S_2 \wedge C_1 \perp C_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2$$

In order to solve these kinds of anomalies, the two PIs should be replaced by another PI i_3 that is a sort of upper bound of the previous ones. In particular, i_3 will be composed by the following fields:

- s_3 is the least upper bound of s_1 and s_2, so that $s_3 \succeq s_1$ and $s_3 \succeq s_2$ hold;
- d_3 is the least upper bound of d_1 and d_2,so that $d_3 \succeq d_1$ and $d_3 \succeq d_2$ hold;
- $C_3 = \{c_{3,i}\}_i$ can be computed as $c_{3,i} = \max(c_{1,i}, c_{2,i})$ where $C_1 = \{c_{1,i}\}_i$ and $C_2 = \{c_{2,i}\}_i$;
- S_3 is the least upper bound of S_1 and S_2 such that $S_3 \succeq S_1$ and $S_3 \succeq S_2$ hold;
- $t_3 = t_1 = t_2$, $G_3 = G_1 = G_2$.

To maintain the semantics of the system, the new PI i_3 should be inserted with the same priority of i_1 (that is $\pi(i_1)$).

Redundancy – $\mathcal{A}_{re}(i_1, i_2)$

A PI i_2 is *redundant* when there is another policy implementation i_1 with a higher priority that matches all the traffic of the first one and its security coefficients are equal to or dominate the other PI's coefficients. The following formula can be used to infer these problems:

$$\mathcal{A}_{re}(i_1, i_2) \Leftrightarrow t_1 = t_2 \wedge s_1 \succeq s_2 \wedge d_1 \succeq d_2$$
$$\wedge S_1 \succeq S_2 \wedge C_1 \succeq C_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2$$

The proposed resolution consists of simply deleting i_2, since it does not add any new semantics to the policy.

Exception – $\mathcal{A}_{ex}(i_1, i_2)$

A PI i_2 is an *exception* of another policy implementation i_1 with a higher priority if they have disjoint security coefficients and i_2 is a superset match of i_1 ($s_1 \prec s_2 \wedge d_1 \prec d_2 \wedge S_1 \supset S_2$) . The corresponding detection formula is:

$$\mathcal{A}_{ex}(i_1, i_2) \Leftrightarrow \pi(i_1) \prec \pi(i_2) \wedge t_1 = t_2 \wedge s_1 \prec s_2 \wedge$$
$$\wedge d_1 \prec d_2 \wedge S_1 \prec S_2 \wedge C_1 \perp C_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2$$

Exceptions are analogous to shadowing anomalies (just the opposite order of precedences), indeed, they share the same solution technique.

Correlation – $\mathcal{A}_{co}(i_1, i_2)$

A PI i_2 is *correlated* with another policy implementation i_1 if they have disjoint security coefficients, i_1 matches some traffic for i_2 and vice versa. In other words, i_1 and i_2 source and destination belong to the same node and there is no other Intra-technology anomaly between policy implementations (i.e. shadowing, redundancy or exception). We can detect these anomalies via the formula:

$$\mathcal{A}_{co}(i_1, i_2) \Leftrightarrow s_1 \not\preceq s_2 \wedge d_1 \not\preceq d_2 \wedge t_1 = t_2 \wedge S_1 \not\preceq S_2$$
$$\wedge G_1 = G_2 \wedge \neg\mathcal{A}_{sh}(i_1, i_2) \wedge \neg\mathcal{A}_{ex}(i_1, i_2) \wedge \neg\mathcal{A}_{re}(i_1, i_2) \wedge i_1 \neq i_2$$

To solve these anomalies, the two PIs i_1 and i_2 can be replaced with a new PI i_3 with the same fields as described in the shadowing anomaly resolution technique. However, the newly created policy implementation will be inserted with a priority $\pi(i_3) = \min(\pi(i_1), \pi(i_2))$.

Inclusion – $\mathcal{A}_{in}(i_1, i_2)$

The PI i_1 *includes* (or dominates) the policy implementation i_2 when all fields of i_1 dominate or are equal to their respective i_2 fields, with the exception of one that is strictly dominant. We can detect these anomalies with the formula:

$$\mathcal{A}_{in}(i_1, i_2) \Leftrightarrow s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge t_1 \succeq t_2 \wedge$$
$$\wedge C_1 \succeq C_2 \wedge S_1 \succeq S_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2$$

The simplest way to solve these anomalies is to delete i_2 (the 'innermost' PI). Though , an administrator can also choose to keep both the policy implementations, by adopting a security in depth approach.

Affinity – $\mathcal{A}_{af}(i_1, i_2)$

A PI i_1 is *affine* with another policy implementation i_2 when they share some fields, but none of the PIs include the other. We can detect these anomalies with the formula:

$$\mathcal{A}_{af}(i_1, i_2) \Leftrightarrow (s_1 \not\preceq s_2 \wedge d_1 \not\preceq d_2 \wedge t_1 \not\preceq t_2) \wedge$$
$$S_1 \not\preceq S_2 \wedge \neg\mathcal{A}_{in}(i_1, i_2) \wedge \neg\mathcal{A}_{in}(i_2, i_1) \wedge i_1 \neq i_2$$

To solve this type of anomalies, a possible solution may be to replace the two PIs should be replaced with a new PI i_3 that is a sort of upper bound of the previous ones:

- s_3 is the least upper bound of s_1 and s_2 so that $s_3 \succeq s_1$ and $s_3 \succeq s_2$ hold;
- d_3 is the least upper bound of d_1 and d_2 so that $d_3 \succeq d_1$ and $d_3 \succeq d_2$ hold;

- t_3 is the least upper bound of t_1 and t_2 so that $t_3 \succeq t_1$ and $t_3 \succeq t_2$ hold;
- $C_3 = \{c_{3,i}\}_i$ can be computed as $c_{3,i} = \max(c_{1,i}, c_{2,i})$ where $C_1 = \{c_{1,i}\}_i$ and $C_2 = \{c_{2,i}\}_i$;
- S_3 is the least upper bound of S_1 and S_2 such that $S_3 \succeq S_1$ and $S_3 \succeq S_2$ hold;
- $G_3 = G_1 = G_2$.

Contradiction – $\mathcal{A}_{co}(i_1, i_2)$

Two PIs i_1 and i_2 are in *contradiction* if their sources/destinations lay on the same node while their technologies are disjoint (that is one PI is using the NULL technology and the other one a security protocol). The formula for detecting these anomalies is:

$$\mathcal{A}_{co}(i_1, i_2) \Leftrightarrow s_1 \not\perp s_2 \wedge d_1 \not\perp d_2 \wedge t_1 \perp t_2 \wedge S_1 \not\perp S_2 \wedge i_1 \neq i_2$$

Due to the high ambiguity of the situation (we cannot distinguish between a 'protect' and a 'do not protect' requirement), the resolution may be the removal of one policy implementation, for instance, the one using the NULL technology.

5.4.3 Network level anomalies

Network level anomalies occur between distinct PIs that belong to different nodes. We can split these anomalies into two main categories: path (cyclic path, monitorability and alternative path anomalies), and channel anomalies (superfluous, filtered channel, L2, skewed channel and asymmetric channel anomalies).

Superfluous – $\mathcal{A}_{su}(i_1)$

A PI i_1 is *superfluous* if it models a tunnel that protects less than all its inner end-to-end channels. That is, the security coefficient of a superfluous tunnel i_1 are smaller than all the encapsulated channels ($\forall\, i_k : s_k \in S_1|_{ip_{src} \times p_{src} \times \ldots} \wedge G_k^* \supset G_1^*$).

This anomalous PIs can be detected by using the formula:

$$\mathcal{A}_{su}(i_1) \Leftrightarrow \nexists\, i_k : s_k \in S_1|_{ip_{src} \times p_{src} \times \ldots} \wedge G_k^* \supset G_1^* \wedge C_k \prec C_1$$

Since all the data transported in the tunnel are better protected than the tunnel itself, the obvious resolution is to delete i_1 (since it is superfluous). However, as in the inclusion anomaly, an administrator could choose to keep the PI to (slightly) increase the security of the network.

Skewed channel – $\mathcal{A}_{sk}(i_1, i_2)$

Two PIs i_1 and i_2 that define two tunnels are *skewed* if their respective channels overlap. This type of anomalies are tricky because the traffic will be sent without any form of encryption in a portion of the network (Fig. 5.4). We can detect these anomalies with the formula:

$$\mathcal{A}_{sk}(i_1, i_2) \Leftrightarrow s_1 \in S_2|_{ip_{src} \times p_{src} \times \ldots} \wedge (|G_1^* \cap G_2^*|) \geqslant 2 \wedge$$
$$\wedge (G_2^* \setminus G_1^* \neq 0) \wedge c_1^c > 0 \wedge c_2^c > 0 \wedge i_1 \neq i_2$$

In order to solve this kind of anomalies, the two PIs must be split in three (or more) non-overlapping policy implementations.

Filtered channel – $\mathcal{A}_{fi}(i_1)$

A PI i_1 is *filtered* when there exists at least one node e in its path with a filtering rule that discards all its traffic. Given a function $\mathcal{F}_e(i_1)$, which returns true if the traffic related to i_1 is dropped and false otherwise, we can formally model this anomaly with:

$$\mathcal{A}_{fi}(i_1) \Leftrightarrow \exists e : e \in G_1 \wedge \mathcal{F}_e(i_1) = true$$

In practice, the output of the function $\mathcal{F}_e(i_1)$ can be populated either by means of a network reachability analysis [78], [100] or by using some firewall policy queries [61].

This anomaly is particularly severe since it completely hinders the connectivity among a number of network nodes. To remove the problem, the administrator can choose to delete the PI i_1 or to modify accordingly the filtering rule.

L2- $\mathcal{A}_{L2}(i_1)$

We have a *L2 anomaly* when a PI that uses a data-link layer technology crosses an area using a different layer 2 protocol. For instance, we have a L2 anomaly when a WPA2 policy implementation crosses some Ethernet nodes, so that we cannot use WPA2 for the whole path. We can express this anomaly by using a function $\mathcal{T}^{(2)}(e)$ that returns the set of technologies at layer 2 supported by the node e. We can then write the formula:

$$\mathcal{A}_{L2}(i_1) \Leftrightarrow \exists e : e \in G_1^* \wedge t_1 \notin \mathcal{T}^{(2)}(e)$$

These anomalies are rather hard to solve since they require a complete edit of the PI, by choosing a technology at a layer that is strictly greater than 2.

Asymmetric channel – $\mathcal{A}_{as}(i_1)$

A PI i_1 is *asymmetric* if no other PI exists with:

1. the source, destination and selectors swapped ($s_1 \not\perp d_2 \wedge d_1 \not\perp s_2 \wedge S_1 \not\perp \overleftarrow{S_2}$);
2. the same technology and security coefficients;
3. the same list of crossed gateways, which are instead in reverse order.

In other words, these problems arise when we have a bidirectional communication with a channel that is weaker than the other. We can identify these anomalies by using the formula:

$$\mathcal{A}_{as}(i_1) \Leftrightarrow \nexists i_2 : s_1 \not\perp d_2 \wedge d_1 \not\perp s_2 \wedge t_1 = t_2 \wedge C_1 = C_2 \wedge S_1 \not\perp \overleftarrow{S_2} \wedge G_1 = \overleftarrow{G_2}$$

The simplest way to solve these anomalies is to ensure that the two PIs have the same security coefficients.[2]

Cyclic path – $\mathcal{A}_{cy}(P^{e_1,e_2})$

There is a *cyclic path* anomaly between two network nodes e_1 and e_2 if there is at least one cycle in the path connecting them. In literature, several algorithms have been proposed to perform cycle detection in graphs in an extremely efficient way [101].

This kind of anomalies can be solved by modifying the PIs in order to remove the cycles.

Monitorability – $\mathcal{A}_{mo}(P^{e_1,e_2})$

A path P^{e_1,e_2} is *monitorable* when there is not an end-to-end channel between e_1 and e_2, which means that, even if the connections are protected by encryption, there is at least one node where an encrypt/decrypt operation is performed, thus potentially breaking the confidentiality of the communication. These anomalies can be detected by using the formula:

$$\mathcal{A}_{mo}(P^{e_1,e_2}) \Leftrightarrow \nexists P^{e_1,e_2} : (|P^{e_1,e_2}| = 1 \wedge i_j \in P^{e_1,e_2} : c_j^c > 0)$$

If the network is not trusted, the obvious way to remove the anomaly is to edit the PIs, so that there are only end-to-end channels between e_1 and e_2 .

Alternative path – $\mathcal{A}_{al}(P_1^{e_1,e_2}, P_2^{e_1,e_2})$

There is an *alternative path* between two nodes e_1 and e_2 if there are two or more different paths that can be taken from the source node to the destination node. These anomalies can be easily found by using the formula:

$$\mathcal{A}_{al}(P_1^{e_1,e_2}, P_2^{e_1,e_2}) \Leftrightarrow \exists P_1^{e_1,e_2}, P_2^{e_1,e_2} : P_1^{e_1,e_2} \neq P_2^{e_1,e_2}$$

[2]Unless the administrator wants this configuration for his network and thus he indicates that the encountered conditions is not a real anomaly.

To remove such a redundancy, administrators have to choose the 'best' path for the communication and delete the remaining ones. The choice can be made by using different strategies, like picking the fastest path or the path containing the PIs with the highest security coefficients.

Chapter 6

Communication Protection Policies: Multi-graph representation

Aiming for a model that also holds practical relevance, we investigated the possibility of a user-friendly representation of the aforementioned anomalies (Chapter 5). It is evident that logical formulas are not easily usable by administrators.

The obvious advantage of such a representation is that it allows a network administrator to visualise the communications at a glance, intuitively identify the anomalies, and immediately see the consequences and the proper reactions.

Starting by the hierarchical view of a network node, already sketched in Section 5.3, we use the multi-graph theory to depict the secure communications and have an equivalent representation of the FOL model described in the previous chapter.

In this chapter, we represent all the CPP anomalies (but those out of place) by using multi-graphs.

6.1 Representation of Policy Implementation

In order to build an anomaly in this graphical representation, firstly we represent the communication end-points by means of the tree-like notation shown in Fig. 5.5, then we have to report the policy implementations.

The policy implementations that enforce end-to-end channels are represented as single directed edges connecting two communication vertices, i.e. the proper communication layer nodes. For instance, in Fig. 6.1, the edge connecting *Layer 3* nodes indicates the IPsec technology. To increase the expressiveness of our representation, each edge is also labelled with both the technology and the security coefficients required by the PI.

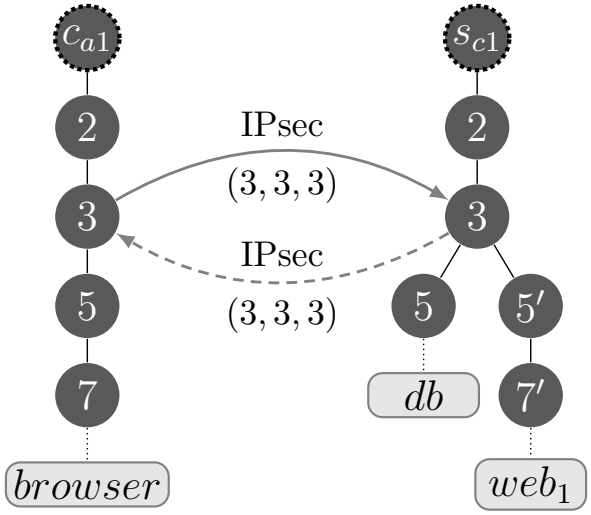


Fig. 6.1 Graphical representation of an end-to-end communication.

To represent the policy implementations that enforce site-to-site and remote-access communications, instead, we add all the network node trees corresponding to the crossed gateways and add a label (next to the technology) with the set of network selectors. Fig. 6.2 reports an example of site-to-site communication.

In the end-to-end communication, for sake of simplicity we do not show the selector tuple S, because S is equal to the all-matching tuple $(*, *, *, *, *, \dots)$.

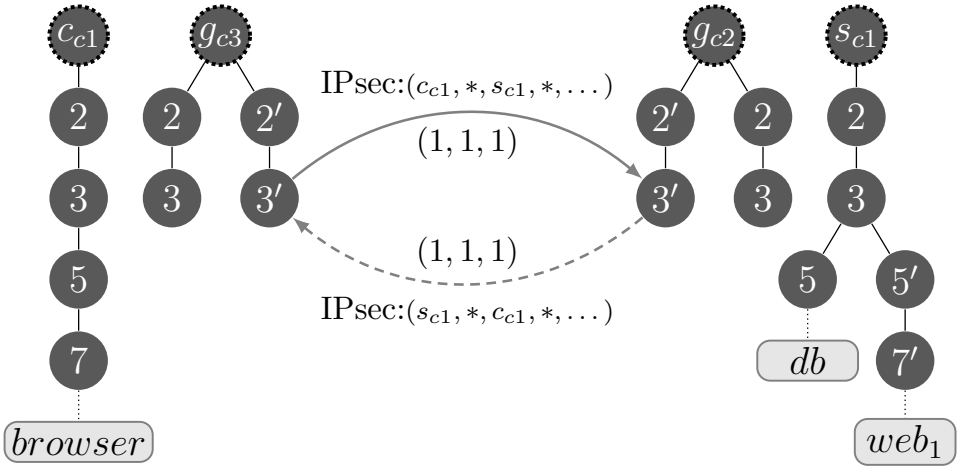


Fig. 6.2 Graphical representation of a site-to-site communication.

The following sections show the graphical representation of the communication protection policy anomaly, using as an example the network scenario in Fig. 5.1.

6.2 PI level anomalies

The classification shown in Fig. 5.6 distinguishes four types of PI level anomalies: out of place, internal loop, non-enforceability and inadequacy.

Out of place

As we have anticipated in the previous paragraphs, out of place is the only anomaly that will not be represented graphically. Visualizing this anomaly negligibly boosts practical usefulness of our graphical representation and significantly increases its complexity.

Internal loop

There is an *internal loop anomaly* when source and destination end-points are on the same node, thus creating a data loop. In Fig. 6.3, source and destination are the same layer 3 node of s_{c1}.

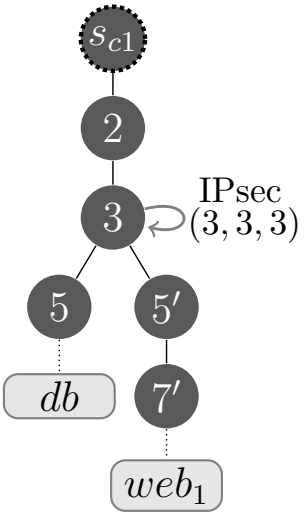


Fig. 6.3 Graphical representation of an internal loop anomaly.

Non-enforceability

There is a non-enforceable PI when PI's technology is not supported by the source or by the destination, or when its security coefficients are 'too high', thus preventing their enforcement. As Fig. 6.4 points out, a non-enforceable PI is represented with an interrupted edge and an over-lined unsupported parameter (technology and/or coefficients).

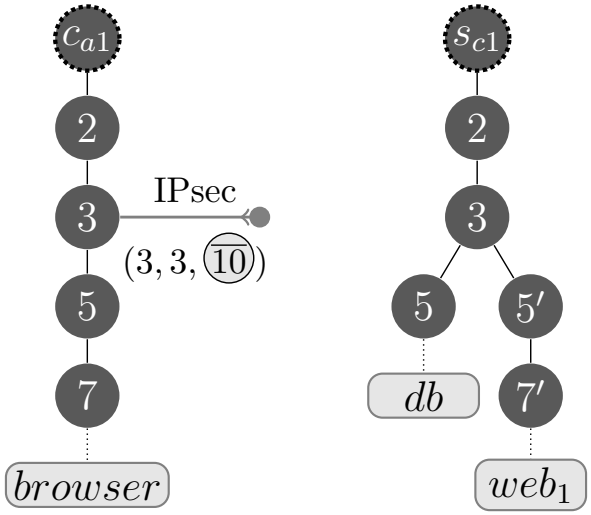


Fig. 6.4 Graphical representation of a non-enforceability anomaly.

Inadequacy

There is an *inadequacy anomaly* when the security coefficients of a policy implementation establish a channel with a security that is lower than an acceptable threshold. As it is shown in Fig. 6.5, an inadequacy PI is represented with an under-lined coefficient below the edge.

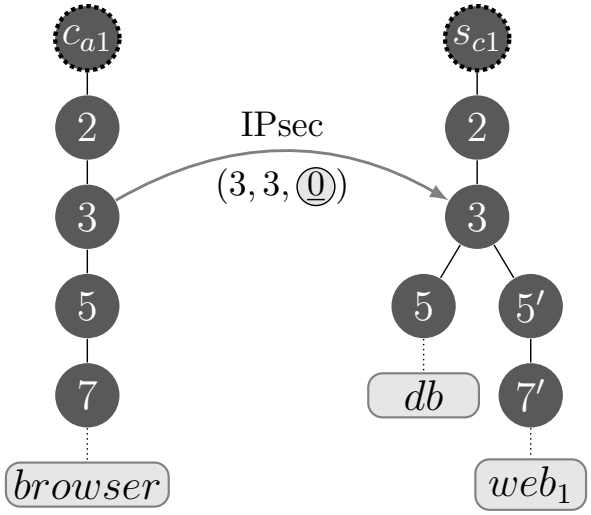


Fig. 6.5 Graphical representation of an inadequacy anomaly.

6.3 Node level anomalies

Node level anomalies occur between two distinct policy implementations lying on the same node. We can distinguish two families of anomalies: *Intra-technology anomalies* and the *Inter-technology anomalies*. Shadowing, exception, redundancy and correlation anomalies are Intra-technology anomalies, while the inclusion, affinity and contradiction anomalies belong to the Inter-technology anomalies.

Shadowing and Exception

As it is reported in Section 5.4.2, the PI i_2 is *shadowed* when there is another policy implementation i_1 –with a higher priority– that matches all the traffic of the first one and has disjoint security coefficients. Furthermore, i_2 is an

exception of another policy implementation i_1 with a higher priority if they have disjoint security coefficients and i_2 is a superset match of i_1. This means that if i_1 is *shadowed* by another PI i_2, the policy implementation i_2 is an exception of i_1.

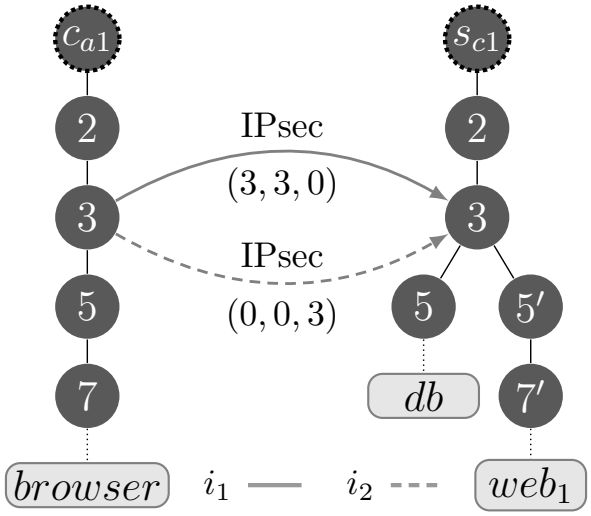


Fig. 6.6 Graphical representation of shadowing and exception anomalies.

Fig. 6.6 offers an example of shadowing and exception anomalies, where the first policy implementation i_1 (solid line) shadows the second policy implementation i_2. In the graphical representation, priority is expressed by the distance from the root node (lower distances meaning higher priority).

Affinity

Two PIs are affine since they share some common aspects, but none of them includes the other one. Fig. 6.7 shows an affinity anomaly between two PIs. The first policy implementation (solid line) enforces IPsec in transport mode and only requires confidentiality, while the second one (dashed line) uses TLS and enforces both payload and header integrity.

An administrator may understand that he can only use (1) either the IPsec channel, if also he adds payload and header integrity, (2) the TLS channel, if he adds confidentiality, or (3) that he can even keep both (provided he adds at least payload integrity to the IPsec channel, that is independently highlighted as an inadequacy anomaly).

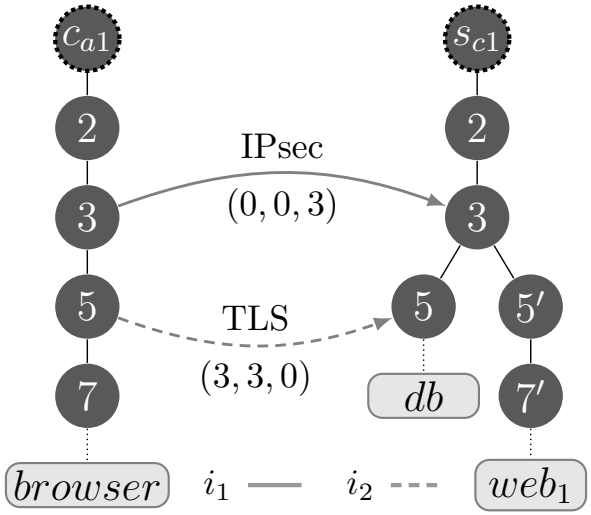


Fig. 6.7 Graphical representation of an affinity anomaly.

Redundancy

A i_2 PI is *redundant* when there is another policy implementation i_1 with a higher priority that matches all the traffic of the first one and its security coefficients are equal or dominate the other PI's coefficients. The following Fig. 6.8 is the representation of redundancy anomalies.

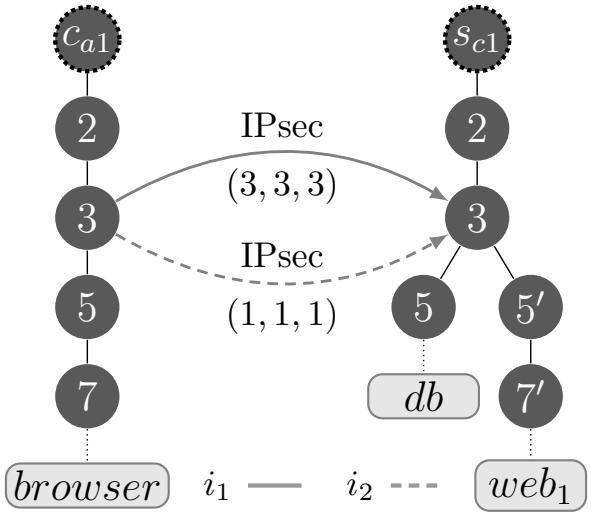


Fig. 6.8 Graphical representation of a redundancy anomaly.

Correlation

The PI i_2 is *correlated* (Fig. 6.9) with another policy implementation i_1, if source and destination of i_1 and i_2 belong to the same node and there is no other Intra-technology anomaly between policy implementations (i.e. shadowing, redundancy or exception). Fig. 6.9 represent the correlation between i_1 e i_2.

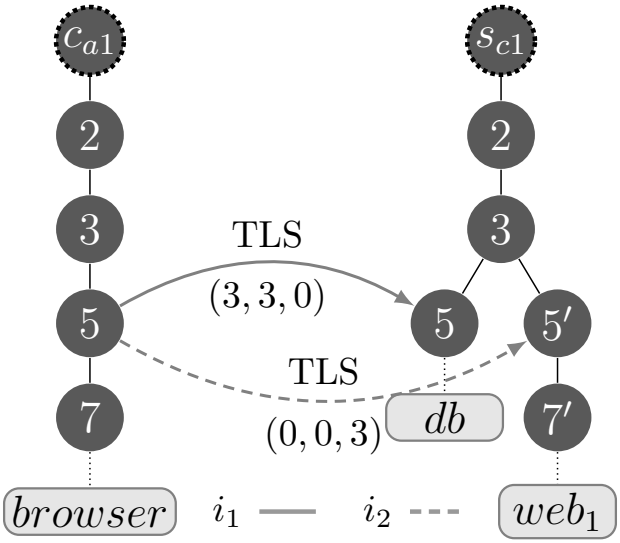


Fig. 6.9 Graphical representation of a correlation anomaly.

Contradiction

Two PIs i_1 and i_2 are in a *contradiction* if their sources/destinations lie on the same node, but one PI uses the NULL technology and the other one specifies a security protocol. Fig. 6.10 represents a contradiction anomaly among two PIs. The first PI (solid line) required NULL technology, while the second one (dashed line) uses TLS and enforces both payload and header integrity.

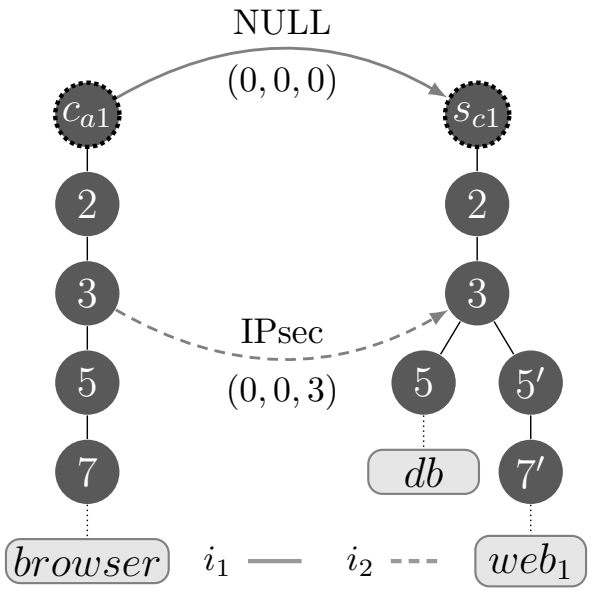


Fig. 6.10 Graphical representation of a contradiction anomaly.

Inclusion

The PI i_1 *includes* (or dominates) i_2 policy implementation when all the fields of i_1 dominate or are equal to their respective i_2 fields, but one, which is strictly dominant. Fig. 6.11 shows an example of graphical representation of an inclusion anomaly.

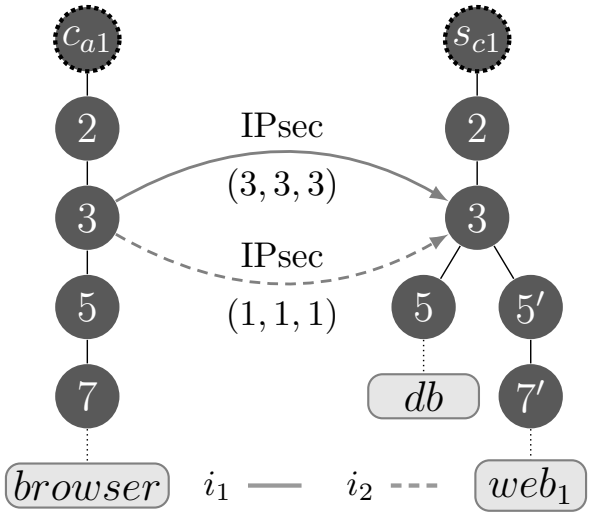


Fig. 6.11 Graphical representation of an inclusion anomaly.

6.4 Network level anomalies

Network level anomalies occur between distinct PIs that belong to different nodes. These anomalies are split in two main categories: path and channel anomalies. Cyclic path, monitorability and alternative path are path anomalies, while channel anomalies are superfluous, filtered channel, L2, skewed channel and asymmetric channel anomalies.

Superfluous

Fig. 6.12 depicts a superfluous anomaly. We recall that a channel is superfluous when another channel covers at least the same traffic, protecting as well the communication with a higher security level. In this case, the IPsec tunnel between g_{c3} and g_{c2} is redundant, so an administrator immediately sees that it can be safely removed.

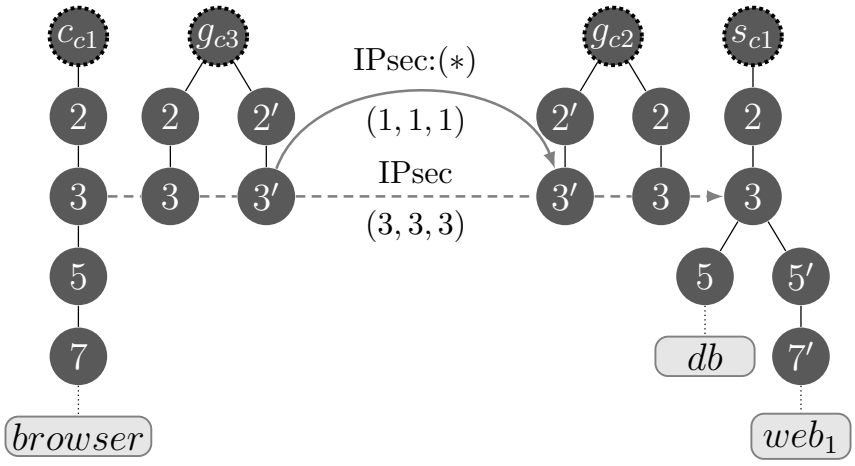


Fig. 6.12 Graphical representation of a superfluous anomaly.

Monitorability

Fig.. 6.13 depicts a Monitorability path anomaly. The multi-graph clearly shows that i_1 and i_2 form a logical connection between C_{b1} and S_{c1} by breaking it into two channels interconnected through G_{c1}. This means that, even if the data is encrypted, in G_{c1} it is possible to spy the traffic of the communication between C_{b1} and S_{c1}.

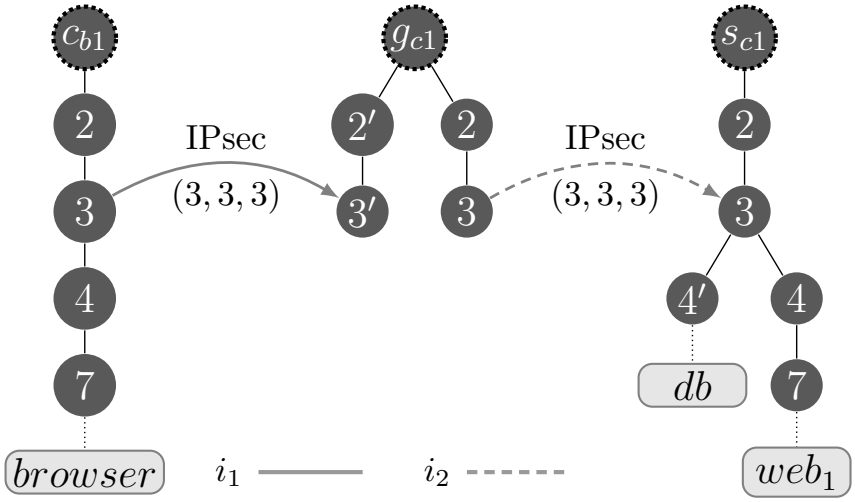


Fig. 6.13 Graphical representation of a monitorability anomaly.

Skewed channel

Two PIs i_1 and i_2 that define two tunnels produce a Skewed channel anomaly if their respective channels overlap. Fig. 6.14 show an example of tunnels overlapped.

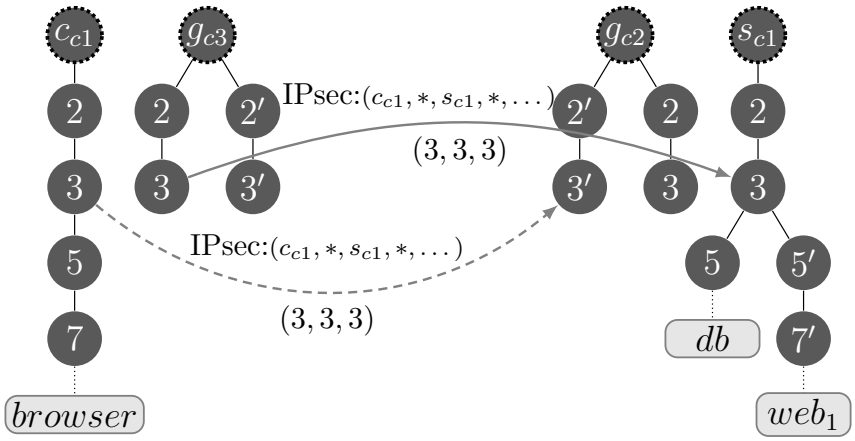


Fig. 6.14 Graphical representation of a Skewed channel anomaly.

Cyclic path

As Fig. 6.15 points out, there is a cyclic path anomaly between two network nodes (c_{c1} and s_{c1}) if there is at least one cycle in the path that connects them.

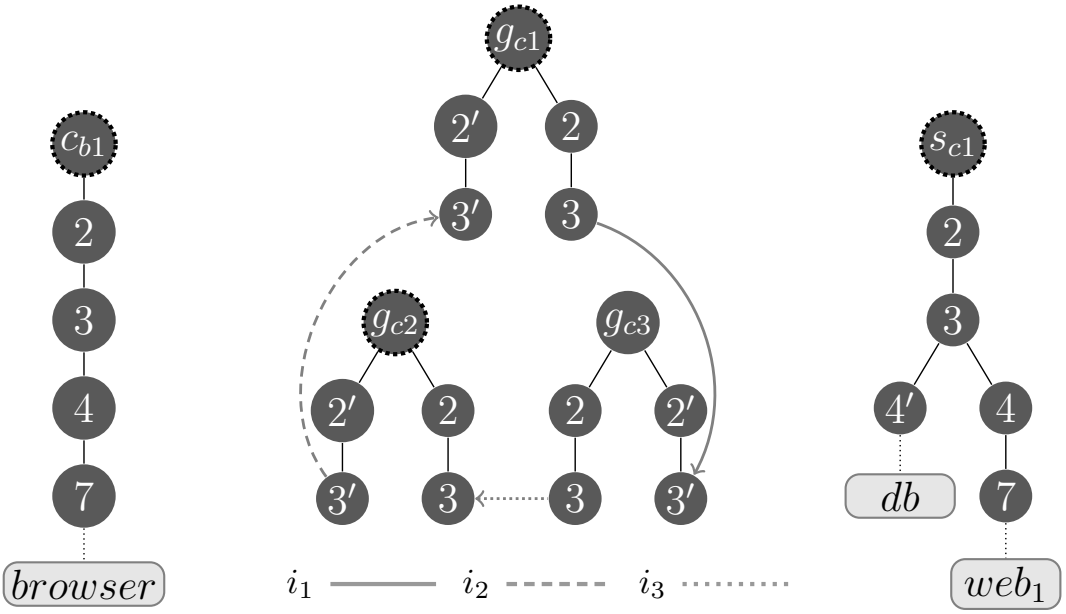


Fig. 6.15 Graphical representation of a cyclic path anomaly.

Filtered channel

A policy implementation is *filtered* when at least one node exists in its path with a filtering rule that discards all its traffic. Fig. 6.16 shows an example of filtered channel anomaly, where the node G_{c2} filtered the communication between c_{b1} and s_{c1}.

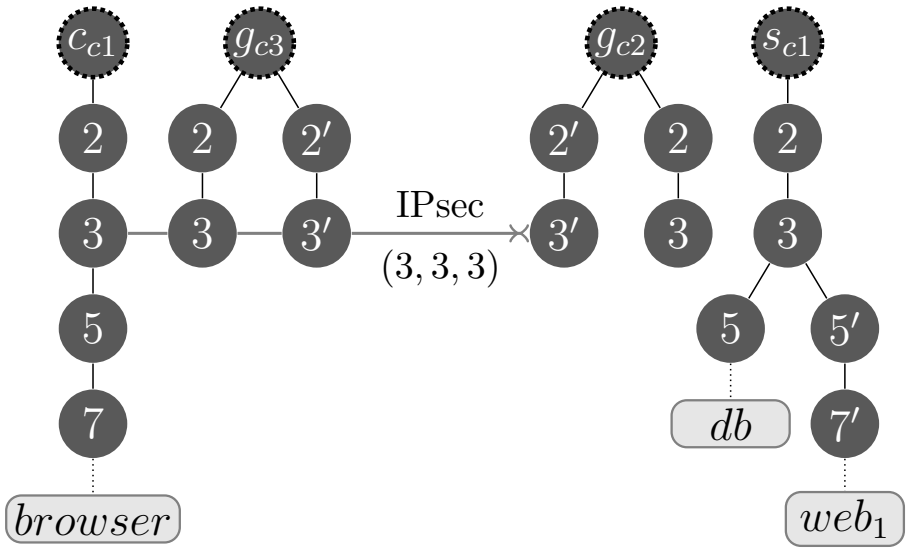


Fig. 6.16 Graphical representation of a filtered channel anomaly.

L2

We have a *L2 anomaly* when a PI that uses a layer two technology crosses an area using a different layer 2 protocol. For instance in Fig. 6.17, we have a L2 anomaly because the WPA2 policy implementation between c_{a1} and c_{a3} crosses the Ethernet nodes g_a1, so that we can not use WPA2 for the whole path.

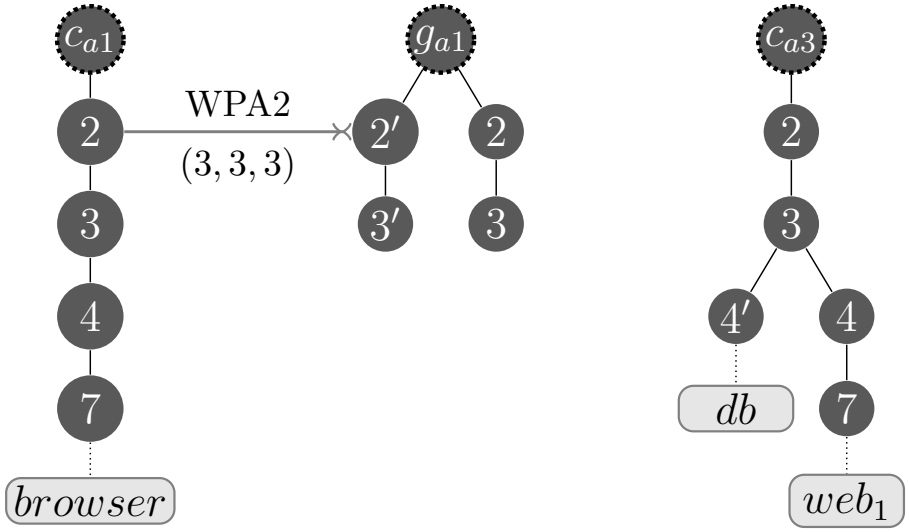


Fig. 6.17 Graphical representation of a L2 anomaly.

Asymmetric channel

There is Asymmetric channel anomaly for a policy implementation if no other PI with the source and destination swapped and the same technology, security coefficients and list of crossed gateways (in reverse order). Fig. 6.18 shows the graphical representation of an asymmetric channel anomaly. In this example the fist PI (solid line) crossed G_{b1} while the second PI (dotted line) crossed G_{b2}.

Alternative path

There is an alternative path anomaly, when there are two or more paths that can be taken from the source node to the destination node. Fig. 6.19 shows

the two paths between c_{b1} and s_{c1}, the first (solid line) crossed g_{c1} while the second (dashed line) crossed g_{c2} and g_{c3}.

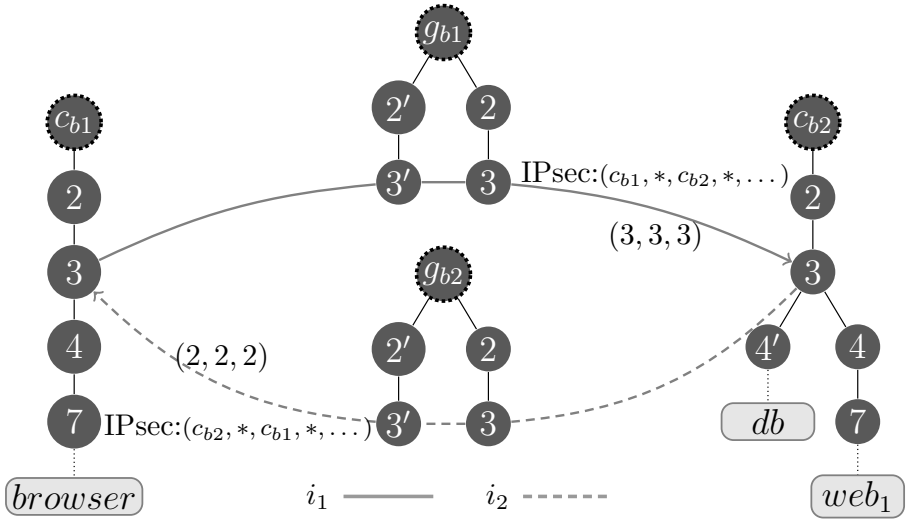


Fig. 6.18 Graphical representation of an asymmetric channel anomaly.

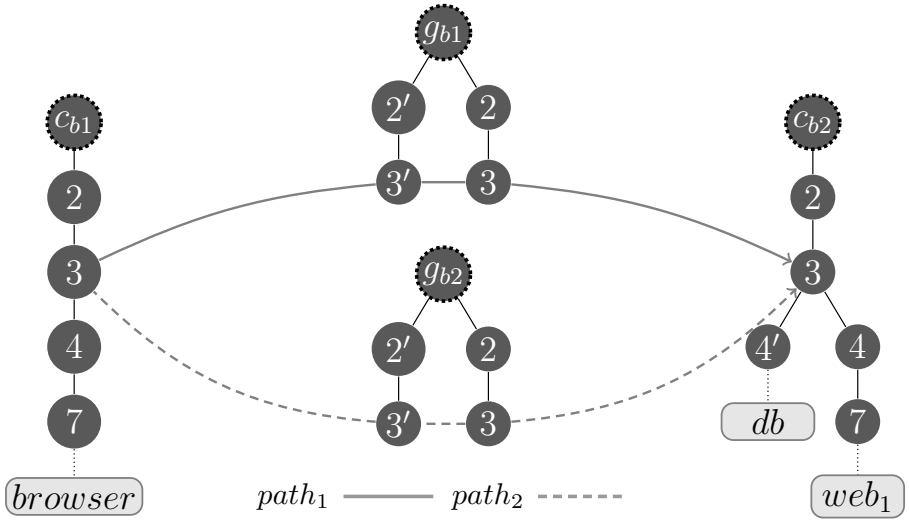


Fig. 6.19 Graphical representation of an alternative path anomaly.

Chapter 7

Communication Protection Policies: Model Validation

In this section, we will present an evaluation of the suitability of our anomaly analysis model. Further details are available in Appendix B.

7.1 Empirical assessment

In order to evaluate the practical importance of our work, we conducted an empirical assessment. We tried to answer two simple yet interesting research questions:

1. Are the anomalies presented in this paper actually introduced by administrators when configuring the CPPs?
2. Does the number of anomalies decrease when administrator expertise grows?

If the first research question RQ1 were confirmed, we could deduce that performing the detection can help improving the policy enforcement correctness in real world networks.

We mainly focused on the new kinds of anomalies presented in this thesis. This is the reason why we did not report statistics on anomalies already present

in literature, namely shadowing, redundancy, exception, correlation, skewed channel (overlapping sessions) and out of place (irrelevances) whose importance has been already proved in other original works [57, 58, 40]. We designed the experiment to be completed by administrators in one hour, therefore we avoided to provide data link information and kept the size of the network reasonably low. This justifies the fact that L2, asymmetric channel, cycle and alternative path anomalies were not considered in our study.

In order to answer the research questions, we conducted an experiment by recruiting a set of 30 subjects. We split them into three categories according to their expertise level (high, medium and low expertise), each one containing 10 people. In the test, we have considered as high-level expert administrators those who have more than two years of experience in the security field, as medium level experts administrators with more than two years of practice in the (non-security) network field and as low level experts the remaining ones.

We asked them to enforce five CPPs (e.g. "all the administrators must securely reach the accounting service") by implementing them as a set of PIs. The landscape was a small network (consisting of 5 subnets, 6 servers, 9 clients and 10 gateways). The network description and the CPPs were available online to the participants both as a web page and as a PDF document to be accessed offline. The participants were asked to write all the PIs where any field was constrained to be valid values (e.g. correct node and protocol names), in order to avoid uninteresting errors. We did not impose neither a time limit nor a maximum number of PIs.

The analysis of the data collected through such an experiment gave us some extremely interesting information. First of all, 93% of the administrators introduced at least one anomaly, regardless of the expertise (Table 7.1). In addition, all the new anomalies have been introduced by at least one administrator (Table 7.2). Interestingly enough, all the anomaly types but contradictions were also introduced by expert administrators. This result successfully answered positively the RQ1 research question, that is the anomalies presented in this paper can appear in real world scenario, hence it is useful to look for them.

The RQ2 research question (the more the expertise of administrators the less the anomalies) has been also confirmed for all the macro-categories of anomalies but one, suboptimal implementations (Table 7.4). Obviously, having

Experience	Insecure communications	Unfeasible communications	Potentialerrors	Suboptimal implementations	At least one type
Low	70.00%	60.00%	60.00%	70.00%	100.00%
Medium	60.00%	30.00%	50.00%	40.00%	90.00%
High	30.00%	20.00%	20.00%	70.00%	90.00%
Average	53.33%	36.67%	43.33%	60%	93.33%

Table 7.1 Percentage of administrators that created at least one anomaly in a macro-category.

Experience	Internal loop	Non-enforceability	Inadequacy	Inclusion	Affinity	Monitorability	Superfluous	Filtered	Contradiction
Low	20.00%	30.00%	40.00%	30.00%	50.00%	30.00%	30.00%	30.00%	30.00%
Medium	10.00%	20.00%	40.00%	20.00%	40.00%	20.00%	30.00%	10.00%	10.00%
High	10.00%	10.00%	10.00%	20.00%	20.00%	20.00%	50.00%	10.00%	0.00%
Average	13.33%	20.00%	30.00%	23.33%	36.67%	33.33%	30.00%	16.33%	13.33%

Table 7.2 Percentage of administrators that created at least one anomaly.

Experience	Internal loop	Non-enforceability	Inadequacy	Inclusion	Affinity	Monitorability	Superfluous	Filtered	Contradiction
Low	1.49%	4.48%	10.95%	2.99%	6.97%	2.99%	7.46%	17.91%	8.96%
Medium	1.50%	3.76%	13.53%	4.51%	5.26%	3.01%	3.76%	9.02%	4.51%
High	1.61%	0.81%	4.03%	3.23%	2.24%	8.87%	8.06%	3.23%	0.00%
Average	1.53%	3.28%	9.83%	3.49%	5.24%	6.55%	4.59%	11.35%	5.24%

Table 7.3 Percentages of anomalies introduced by administrators.

Experience	Insecure communications	Unfeasible communications	Potential errors	Suboptimal implementations	Total
Low	18.41%	22.39%	15.92%	7.46%	64.18%
Medium	16.54%	12.78%	9.77%	9.77%	48.87%
High	12.90%	4.03%	2.42%	12.90%	32.26%
Average	16.38%	14.63%	10.48%	9.61%	51.09%

Table 7.4 Percentages of anomalies introduced by administrators grouped in macro-categories.

a better understanding of a network and its different security controls, reduces the chance of introducing anomalies. This is particularly evident for the filtered anomalies, as administrators also have to consider interactions with firewalls to avoid them, but it is also valid for the non-enforceability, inadequacy, affinity, and contradiction anomalies (Table 7.3).

On the other hand, suboptimal implementations tend to increase, because expert administrators add more superfluous anomalies than the inexpert ones. This is due to the fact that highly experienced administrators tend to add several levels of protection to the communication (defense in depth), although this was not expressly required in the exercise. Moreover, expert administrators' PIs also contain several monitorability anomalies, since they tend to make an extensive use of tunnels while the less experienced ones mainly use end-to-end channels.

In short, experienced administrators tend to break secure communications to improve overall network performances. In this sample network, monitorability anomalies are not the most serious issues (as we had homogeneous security levels in all the networks). Nonetheless, it is certainly better to double check them. Finally, there are a number of internal loop anomalies, probably due to mere distraction errors, that are constant regardless of the expertise level.

To assess the statistical significance of the results split into three expertise levels (high, medium and low), we performed an analysis of variances (ANOVA) with a significance level of 0.05 (the most commonly used threshold).

We assume that a hypothesis is valid if the significance level is less than 0.05 (the most commonly used threshold).

ANOVA tests the non-specific null hypothesis (H_0) that all three population means are equal, that is:

$$H_0 : \mu_{higt} = \mu_{medium} = \mu_{low}$$

On the other hand, the alternative hypotheses is obviously:

$$H_a : H_0 \text{ is false}$$

We performed the ANOVA analysis on two different data sets:

- the number of anomalies introduced by each administrator (test 1);
- the number of anomaly types introduced by each administrator (test 2).

The computed P-values of the ANOVA are 0.034 (test1) and 0.006 (test2). In both cases, being the P-values less than the significance level, the null hypothesis can be rejected, thus the population means are not all equal when introducing anomalies. In addition both the F statistic numbers are greater than their minimum F critical values, further contributing to accepting the alternative hypothesis [102].

In the following sections we report more details about our analysis.

7.1.1 Test 1 – anomalies for each administrator

In Table 7.5 we present the number of anomalies introduced by each administrator depending on their expertise. Administrators have been divided in l_1-l_{10}, m_1-m_{10} and h_1-h_{10}. Each cell value represents the number of anomalies introduced by an administrator.

Table 7.6 shows the statistical results computed on these data, while Table 7.7 details the ANOVA computation values [102]. The table columns have the following meanings:

- SS stands for 'Sum of Squares' and it is the sum of squared deviations;
- df is the acronym for 'Degree of Freedom';
- MS is the 'the Mean sum of Squares due to the source';
- F is the 'F statistic', a value used to accept the alternative hypothesis if it is large enough;
- P-value is the 'Probability' of having an F statistic large enough such that the null hypothesis is true;
- F crit is the 'F critical value', used to accept the alternative hypothesis if F statistic is greater than this threshold.

7.1.2 Test2 – anomaly types for each administrator

Table 7.8 presents the number of anomaly types that has been triggered: an anomaly type has been considered if at least one anomaly of that type has been introduced by the administrators. We indicated how many anomaly types have been introduced for each expertise level. Tables 7.9 and 7.10 show the statistical results computed on these data.

	high expertise administrators									
admin	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}
anomaly #	1	2	2	4	10	7	0	2	5	7

	medium expertise administrators									
admin	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
anomaly #	6	7	6	2	3	0	7	6	6	22

	low expertise administrators									
admin	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
anomaly #	4	20	24	4	20	19	13	4	9	12

Table 7.5 Anomalies count per administrators (test 1).

expertise	admins	anomalies sum	average anomalies	variance
high	10	40	4	10.222
medium	10	65	6.5	35.167
low	10	129	12.9	57.211

Table 7.6 Statistics for test 1.

source of variation	SS	df	MS	F	P-value	F crit
between groups	421.4	2	210.7	6.161	0.006	3.354
within groups	923.4	27	34.2			

Table 7.7 ANOVA for test 1.

	high expertise administrators									
admin	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}
anomaly #	1	1	1	2	2	2	0	2	3	2

	medium expertise administrators									
admin	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
anomaly #	1	3	1	2	2	0	3	2	3	4

	low expertise administrators									
admin	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
anomaly #	3	4	4	3	1	3	3	1	3	4

Table 7.8 Anomaly types count per administrators (test 2).

expertise	admins	anomaly types sum	average type	variance
high	10	16	1.6	0.711
medium	10	21	2.1	1.433
low	10	29	2.9	1.211

Table 7.9 Statistics for test 2.

source of variation	SS	df	MS	F	P-value	F crit
between groups	8.6	2	4.3	3.84	0.034	3.354
within groups	30.2	27	1.119			

Table 7.10 ANOVA for test 2.

7.2 Complexity analysis

We will now derive some complexity formulas that prove the theoretical performance of our model. We will start with a simple observation. Our approach can be split in two consecutive phases. The first one is a *pre-computation phase*, where the tree representation of the network and its paths are obtained. The second one is an *analysis phase* that consists of the real anomaly detection pass.

Let's suppose that we have a network consisting of $\mathcal{E}$ entities (IPs, ports, addresses, ...), $\mathcal{I}$ policy implementations and $\mathcal{C}$ connections between the network entities created by the PIs (obviously $\mathcal{C} \geq \mathcal{I}$).

To create the tree representation of the network nodes, we need to check every single entity, so that this process has an exact complexity of $\mathcal{E}$. Finding all the simple paths[1] in an acyclic graph is a NExpTime problem with a maximum complexity of $O(e^{\mathcal{E}})$. Note, however, that real networks are scarcely connected and that multiple paths between two different nodes are quite rare, making these calculations also feasible in large IT infrastructures. In addition, an administrator can choose to limit the number of paths to check some fixed value $\mathcal{P}$, typically $\mathcal{P} \lll e^{\mathcal{E}}$. Hence, the total complexity of the pre-computation phase is $\mathcal{E} + \mathcal{P}$.

Regarding the analysis phase, we have to take into account the distinct characteristics of anomaly detection formulas. In particular, we have that:

- internal loop, out of place, non-enforceability, inadequacy, filtered channel, L2 and asymmetric channel anomalies algorithms work on a single PI at a time, so that they have a complexity of $\mathcal{I}$;
- shadowing, redundancy, exception and inclusion anomalies algorithms needs an ordered pair of PIs, thus having a complexity of $\mathcal{I}(\mathcal{I}-1)$. Superfluous anomalies as well have a quadratic complexity since they needs to test every PI against all the other ones;
- correlation, affinity and skewed channel anomalies algorithms work on unordered pairs of PIs, giving a complexity of $\mathcal{I}(\mathcal{I}-1)/2$;

[1] A simple path is a path with no duplicate vertices.

- monitorability and alternative path anomalies algorithms work on every path, hence they have a complexity of $\mathcal{P}$;
- cyclic path anomaly algorithm can be efficiently performed using a proper cycle detection algorithm such as [101], that has a complexity of $O(\mathcal{E}+\mathcal{C})$. Note that its complexity is not necessarily $\mathcal{P}$ since a graph with some loops has an infinite number of paths.

Summarising, the total complexity of the analysis phase is:

$$\mathcal{I}+\mathcal{I}(\mathcal{I}-1)+\mathcal{P}+O(\mathcal{E}+\mathcal{C}) \approx \mathcal{I}^2+\mathcal{P}+O(\mathcal{E}+\mathcal{C})$$

7.3 Performance analysis

We implemented our anomaly detection model and tested it in several scenarios using a number of synthetically generated networks in order to assess its running time.

Our tool was developed using Java 1.8 and exploited the natural graph-based representation of ontologies offered by OWL API 3.4.10 and the reasoner Pellet 2.3.1. We performed all our tests on an Intel i7 @ 2.4 GHz with 16 GB RAM under Windows 7.

Each test was performed on several ad-hoc scenarios consisting of an automatically generated network with a parametric structure where we could specify: 1) the number of network entities; 2) the number of policy implementations; 3) the percentage of conflicting PIs. We chose to fix the number of conflicting PIs at about 50%, since, from our empirical analysis, on average, an administrator only writes about half of the policy implementations without any kind of conflict (Table 7.4). We performed two main kinds of tests. In the first one, we fixed the number of network entities and increased the number of PIs, while in the second one, we did the opposite (fixing the number of PIs and changing the entities count). Fig. 7.1 shows the test results when fixing the number of PIs respectively at 100, 250 and 500, while Fig. 7.2 shows the graphs when the network entities count is 100, 250 and 500. For each test, we kept track of three times: pre-computation phase (dotted lines), analysis phase (dashed lines) and total times (the solid lines).

Our tool proved to be very scalable, achieving a total time of less than two minutes in the worst scenario (500 PIs and 500 network entities). In addition, the results are aligned with the complexity analysis discussed in Section 7.2. For instance, by increasing the number of network entities (Fig. 7.1), we may observe that the times tend to grow, while by fixing the number of entities (Fig. 7.2) the pre-computation phase time is completely unaffected.

All the tests presented in this Chapter were conducted on a set of synthetically generated networks. Note that the generated scenarios do not provide a fully specified network. The output network graph just includes the minimum information used by our anomaly classification algorithms to work (e.g. no middleboxes, not all the connections between network nodes). Our generation algorithm is parametric and takes as input three arguments: 1) the number of non-conflicting PIs n_{PI}; 2) the number of conflicting PIs $\overline{n}_{PI}$; 3) the number of network entities n_e.

Our approach works in three sequential steps:

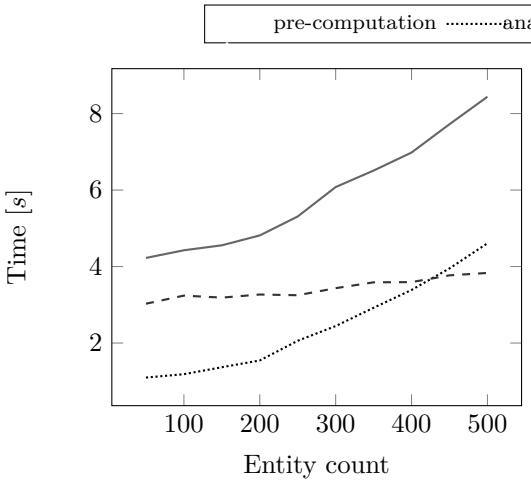
1. generation of n_{PI} non-conflicting PIs. This phase randomly generates policy implementations until the desired number of PIs is produced. Depending on the type of secure communication types, three different procedures are implemented:

    - end-to-end scheme: a new client and a new server are randomly generated and added to the network graph. The connection endpoints are then used to form a single PI that connects them with randomly chosen non-NULL end-to-end technology;
    - site-to-site scheme: a new client, a new server and $n_g \geq 2$ gateways are randomly created and added to the network graph. A set of edges are added to the network graph to properly connect the client and the server with the close gateways and to connect the gateways in the proper order. Then, the algorithm creates a PI connecting the client to the server (which may use the NULL technology) and $n_g - 1$ PIs to form $n_g - 1$ tunnels that connect pairs of adjacent gateways. The selector of the intended client-server traffic was associated to the PI;

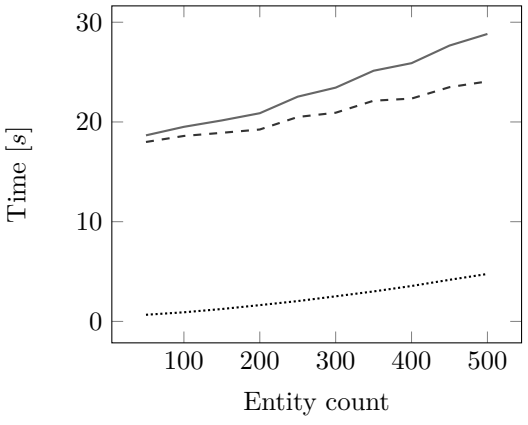
- remote-access scheme: a new client, a new server, and $n_h \geq 1$ gateways are randomly created and added to the network graph nodes. Then, the algorithm creates a PI connecting the client to the server (which may use the NULL technology) and n_g PIs to form n_g tunnels from the client to the last gateway. The selector of the intended client-server traffic was associated to the PI.

2. generation of $\overline{n}_{PI}$ conflicting PIs. This phase randomly generates the policy implementations to introduce the anomalies. The algorithm first picks with the same probability one of the nineteen anomalies in our model. Then it selects zero or more network entities generated during the previous step (the exact number depends on the chosen anomaly) and, if needed, it randomly generates the required entities. Finally, it creates the PIs accordingly;

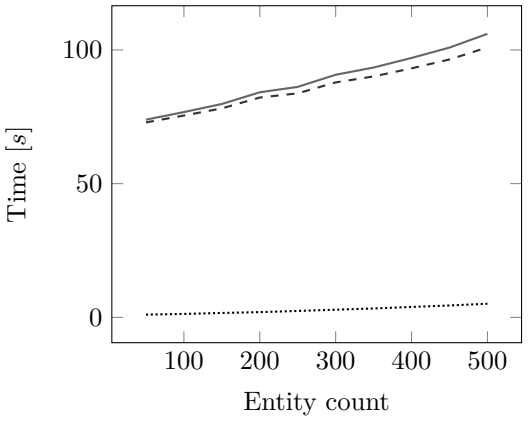
3. network graph completion. During this phase, the algorithm first randomly generates the network entities until the limit n_e is reached. Note that this phase might be skipped if all the previous phases have already generated all the required network entities. Subsequently, the algorithm completes the network adding a bare minimum set of network connections, if necessary. Note that, in the general case, a full network connectivity is not required, and hence not computed, since we are only interested in computing the analysis time of our implementation and adding such additional data does not impact our tool's performance tests.



(a) With 100 PIs.

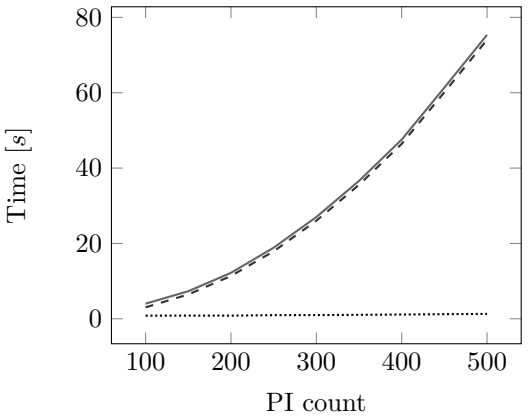


(b) With 250 PIs.

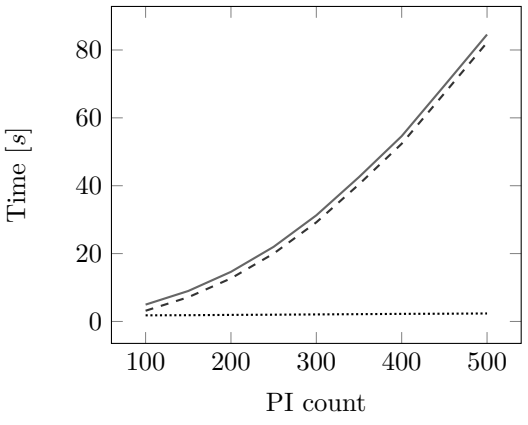


(c) With 500 PIs.

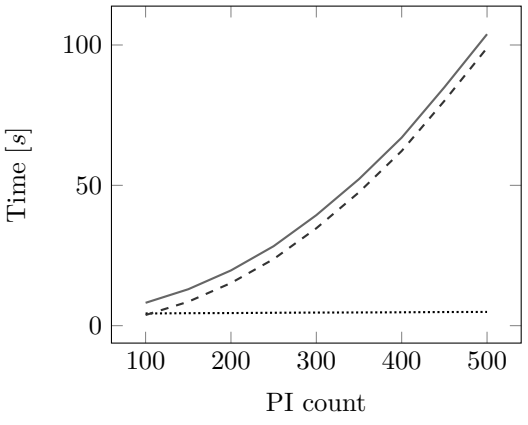
Fig. 7.1 Time to perform the anomaly analysis of a fixed number of PIs depending on the number of entities.



(a) With 100 entities.



(b) With 250 entities.



(c) With 500 entities.

Fig. 7.2 Time to perform the anomaly analysis on networks of a fixed size depending on the number of PIs.

Unified Model for Policy Analysis

Chapter 8

Unified Policy Analysis: Model Definition

We recall that, the complexity of the network topology together with the heterogeneity of network services make the network security policy specification a hard task, even for skilled and experienced administrators.

In this chapter, we presents the UMPA model (Unified Model for Policy Analysis) for detecting network conflicts and irregularities, in order to avoid erroneous and unexpected network behaviours. As we have already mentioned, the literature has proposed approaches of anomaly analysis applied into a single policy domain (*Intra-domain* analysis), while the UMPA model aims at embracing many policy domains in a single analysis process by performing what we name *Inter-domain* policy analysis.

8.1 Motivating Example

In this section, before formally describing the concepts of the UMPA model, we present an example to informally introduce our work.

We use as a reference the simplified network scenario depicted in Fig. 8.1, where different users connect via remote access to the corporate data center. Some services are used by all the corporate users and they are located in the "Global" subnet, which includes the mail server and an anti-spamming

functionality. Other services are department-specific and they are located in the Department1-3 subnets. Department1 offers an internal repository and a database, Department2 provides an application service, while Department3 has three file servers (e.g. FTP servers) managed by a service of intrusion detection.

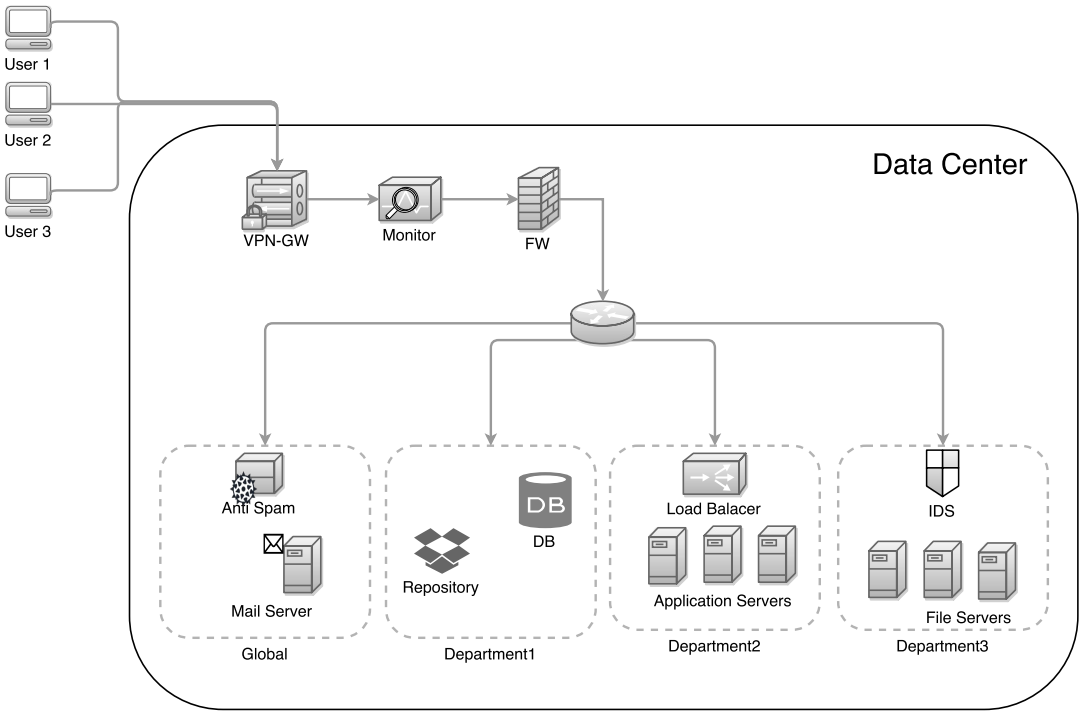


Fig. 8.1 Reference example of a network scenario.

In this example, a user is authorized to access the corporate mail server and the services available for the department he belongs to. To implement this policy, user traffic has to pass through a set of security controls before accessing the different services. Indeed, at the border, there are a VPN gateway, a traffic monitor, a firewall and a router. These security functions process the ingress traffic exactly in this order, dropping or altering the traffic by forwarding (i.e. the firewall) and modifying the headers and/or payloads of the packets received (i.e. the VPN gateway). They can also log traffic info (i.e. the monitor).

Moreover, there are other high-level requirements to implement:

i) drop all the encrypted traffic in outbound to prevent any information disclosure;

ii) permit access to authenticated users;

iii) use a load balancer to access the Department2 applications;

iv) supervise the access to the file servers of Department3 (through an IDS).

Although the proposed scenario is small and simple, we can find some Inter-domain anomalies (defined in Chapter 4). Let us assume the following authorized activities:

- User_1 wants to connect via SSH to the Department1 DB;
- User_2 wants to access the Department2 application service;
- User_3 wants to download a file from the Department3 file server.

In this case, an administrator may notice four incorrect implementations of the security requirements:

1. the monitor cannot analyse the traffic towards the DB because it is encrypted;
2. User1 does not receive responses from DB because the firewall drops the traffic;
3. the monitor cannot distinguish the three different application servers because the load balancer rewrites the source IP addresses;
4. the IDS configuration depends directly on the firewall configuration.

These inconsistencies can be detected by means of the UMPA model, which is able to identify the anomalies by analysing the policies of the security functions in the network. In particular, we focus on the modification induced by the function into the packets.

For each security function, it is necessary to specify its capabilities (in terms of the actions it can apply to the packets) and the configuration rules, i.e. its policy. The model outputs the anomalies as a set of rule pairs that need to be inspected by an administrator. The anomalous rules may belong to different security functions (i.e. Inter-domain anomalies).

In the next sections, we will show how our model enables the discovery of these anomalies. Informally, for anomalies (1) and (3), the monitor, a

function that only reads information in the packet, has a rule to log packets that are received after another function has modified them. In the case (1), the modification is performed by the VPN gateway, in the case (3), by the load balancer. For anomaly (2), it is the rule to drop all the encrypted traffic that prevents replies from reaching User2. Our model detects that there is a filtering rule dropping packets that are transformed (i.e. encrypted). Analogously, for anomaly (4), the model discovers that the IDS has to look into traffic that is filtered by another function.

8.2 Modelling Approach

In this section, we describe the UMPA model, which is able to both represent network policies of different domains and to detect the Intra- and Inter-domain anomalies among such policies. In order to perform those targets, the model of a network security policy is composed of four sets, that are:

Network fields
The *network fields* are atomic elements that identify information a network policy needs to keep track. In others word, the network fields are representations of the necessary state and/or prerequisites that define whether a policy rule actions should be performed. When the network fields associated with a policy rule evaluate to TRUE, then (subject to other considerations such as rule priorities and decision strategies) the rule should be enforced.

Examples of such network fields (but they are not limited to) could be the packet headers. Other information (e.g. network node ID, traffic label, cipher algorithm etc...) could be needed to designate the events and conditions an administrator wants to manage.

Policy actions
The *policy actions* are a set of atomic elements that represent either the real action performed by a network node (e.g. a firewall is configured to *deny* or *allow* a packet under certain conditions), or the parameters and information that characterize that action (e.g. algorithm, technologies, protocols to use);

Policy Implementation

We recall that the *Policy Implementation* (*PI*) is a data-structure to pinpoint in a formal and abstract way the policy rules enforced by a network node for a certain domain. This means that the *PI* data-structure must be defined so that the *PI*:

i) identifies the condition and events that administrators want to manage through conditions expressed on the network fields;

ii) knows the policy actions that describe the way those events are managed;

iii) is designed to be applied to a specific policy domain.

In the UMPA model, we have extended the (*PIs*) that we have presented in Chapter 5. In particular, we raise the level of generalization of the *PI* structure in order to be able to map many policy domains in a single model.

Detection rules

The *detection rules* are a set of conditions that distinguish the possible anomalies among *PIs*. For a particular policy domain, it is possible to exploit the existing works on the policy analysis for that domain. Otherwise, an administrator could define his set of *PI* anomalies, either Intra-domain and Inter-domain.

Now we start to analyse in depth the *PI* structure and then the *PI* anomalies.

8.2.1 PI structure

A *PI* is composed of a sequential set of network fields (n) and a set of policy actions (a):

$$pi_i = (n_{i1}, n_{i2}, ..., n_{im}, a_{i1}, a_{i2}, ..., a_{im})$$

Actually, a *PI* has a different set of network fields and policy actions, based on the domain where the *PI* is defined. In addition, among the network fields and policy actions of the *PIs*, a set of relations $\mathfrak{R}$ must be defined in order to establish the *PI* anomalies. In detail, the proposed model supports four relations $\mathfrak{R}$ between network fields (the same relations can be defined for the

policy actions), which are satisfied based on the real values taken by the network fields (or actions) involved in the relations:

- **equivalence**: the k-th network fields of two policy implementations (pi_i and pi_j) are equivalent (or equal) if they have exactly the same value;

$$n_{ik} = n_{jk}$$

- **dominance**: the k-th network field of pi_i dominates the k-th network field of pi_j if it is a generalization of the latter. For example, n_{i1} is the IP addresses 1.1.*.* and n_{j1} is the IP address 1.1.1.*, in this case n_{i1} dominates n_{j1};

$$n_{ik} \succ n_{jk}$$

- **correlation**: two network fields (n_{ik} and n_{jk}) of two policy implementations (pi_i and pi_j) are correlated if they share some common values, but none of them includes (or dominates) the other one. For example, if n_{i1} and n_{j1} are port number and n_{i1} ranges from 1 to 75, while n_{j1} from 50 to 100, then they are correlated because the range [50, 75] is shared by both fields;

$$n_{ik} \sim n_{jk}$$

- **disjointness**: two network fields (n_{ik} and n_{jk}) are disjoint if they do not share any value. On the other hand, if a network field is equivalent, correlated or dominates another, those fields are *not-disjoint* ($n_{ik} \not\perp n_{jk}$).

$$n_{ik} \perp n_{jk}$$

Note that a network filed (or a policy action) may not support one of the aforementioned relations, due to the value type it takes (e.g. integer, IP address, boolean, enumeration and more). For example, a port number can be equal to another one, but it cannot dominate a range of port numbers. Instead a range of port numbers can dominate a single value of port number.

The proposed set of relations allows the model to achieve high-level flexibility and generality in order to:

1. extend to more domains;

2. detect anomalies among policy rules of different domains;

3. enrich the set of policy anomalies (Inter- and Intra-domain) by allowing administrators to define their own set of anomalies.

In fact this set of relations $\mathfrak{R}$ gives the means to impose conditions on *PI* elements, whose value types are not known a-priori.

8.2.2 PI anomaly detection rules

The *PI* anomalies (A) are defined in form of *detection rules*. The detection rules are, in turn, defined according to the existing relations $\mathfrak{R}$ among the network fields and policy actions of a *PI*.

In the proposed model, the detection rules are based on the FOL and are expressed using Horn clauses. Horn clauses are frequently encountered in model theory because they exhibit a simple and natural rule-like form. Also, these clauses can be easily translated in many different logic programming languages, such as Prolog, or generic programming language such as C or Java. In particular, the Horn clauses can be simply used to represent all the axioms used in the proposed model, expressed in the form of disjunction of literals (clauses) with at most one positive literal:

$$\neg C_1 \vee \neg C_2 \vee \cdots \vee \neg C_n \vee I$$

Alternatively, they can be expressed in a more natural way as a set of positive conditions implying an assertion and this is the form chosen in our model:

$$C_1 \wedge C_2 \wedge \cdots \wedge C_n \Rightarrow I$$

In our model, every clause is the relation between network fields (e.g. n_k) or policy actions (e.g. a_k) of one or more *PIs* (e.g. pi_i and pi_j), also belonging to different domain, like this example:

$$C_1 := n_{ik}\ \mathfrak{R}\ n_{ih},\ C_2 := n_{ik}\ \mathfrak{R}\ n_{jh}$$
$$C_3 := a_{ik}\ \mathfrak{R}\ a_{ih},\ C_4 := a_{ik}\ \mathfrak{R}\ a_{jh}$$

where n_{ik} and n_{jh} (or a_{ik} and a_{jh}) identify two generic network fields (or policy actions) in the ordered sequence of network fields (or actions) in the PI structure.

Hence a generic form of a detection rule can take a form like the following, but they are not limited:

$$n_{iq} \,\mathfrak{R}\, n_{jq} \;\wedge ... \wedge\; n_{ik} \,\mathfrak{R}\, n_{jh} \;\wedge ... \wedge\; a_{ik} \,\mathfrak{R}\, a_{jh} \;\Rightarrow A(pi_i, pi_j)$$

where $A(pi_i, pi_j)$ is the anomaly triggered by the policy implementations pi_i and pi_j.

Furthermore, the Horn clauses support the also the Universal ($\forall$) and Existential ($\exists$) quantifier:

$$\forall x|(\neg C_1 \vee \neg C_2 \vee \cdots \vee \neg C_n \vee A)$$
$$\exists x|(\neg C_1 \vee \neg C_2 \vee \cdots \vee \neg C_n \vee A)$$

where x consists of variables appearing in I and in the B_j .

Hence examples of detection rules can take a form like the following:

$$\forall n_{iq}, n_{jq}| \wedge n_{iq} \,\mathfrak{R}\, n_{jq} \;\wedge ... \wedge\; n_{ik} \,\mathfrak{R}\, n_{jh} \;\wedge ... \wedge\; a_{ik} \,\mathfrak{R}\, a_{jh} \;\Rightarrow A(pi_i, pi_j)$$
$$\exists n_{iq}, n_{jq}| \wedge n_{iq} \,\mathfrak{R}\, n_{jq} \;\wedge ... \wedge\; n_{ik} \,\mathfrak{R}\, n_{jh} \;\wedge ... \wedge\; a_{ik} \,\mathfrak{R}\, a_{jh} \;\Rightarrow A(pi_i, pi_j)$$

In addition in our model we extend the definition of Clause with these axioms:

- a set of condition is also a condition $C_1 = C_2 \wedge C_3 \wedge ... \wedge C_n$
- the negation of a condition is also a condition $C_1 = \neg C_2$

Having provided an overview of how we model the network security policies and the anomalies in the UMPA model, we now move to apply our model to the use-case of anomaly analysis.

Chapter 9

Unified Policy Analysis: Model Validation

In this chapter, we analyse three policy types: communication protection, filtering and traffic flow policies. For each of them, using the aforementioned UMPA model, we present an example of possible *PI* structure and a definition of detection rules that represent the Intra-domain and Inter-domain policy anomalies.

For the filtering policies, we use the main important work in this domain proposed by *Al-shaer et al.* [8].

While for the second case study of the communication protection policies, we refer to our study on the analysis presented in this thesis (Chapter 5). As concerning the latter, a new definition of *PI* structure and detection rules is proposed, in the domain of flow-based policy.

We have considered these three case-studies to prove the generality and flexibility of our model, because it is able to cope with a variety of scenarios, like case-studies already known in the literature or new examples of policies both security-related and not.

Finally in the last section of this chapter, we report a fist approach Inter-domain policy analysis among the three policy types used as use case.

9.1 Packet Filter Policy

In order to validate the usability and expressibility of this model, we use as example the well-known *Packet Filtering Policies* (PFPs) domain.

PFPs are used in a single- or multi-firewall environment to defend secured networks by filtering unwanted or unauthorized traffic from or to the secured network. For an administrator, it could become difficult to correctly deploy a large number of PFPs in a network with many firewalls and easy to trigger packet filter.

Therefore a PFP anomaly analysis is needed and it can be based on a PIs structure composed of eight network fields:

$$pi_{fp} = (f, r, ip\_src, ip\_dst, t, p\_src, p\_dst, a)$$

where:

- f is an incremental firewall identifier to reflect the order in which the received traffic is processed by a sequence of firewalls;
- r is an incremental firewall rule identifier, valid within the firewall identified by f;
- *ip_src* and *ip_dst* are respectively the source and destination IP addresses of the traffic , can take as possible values a single IP (e.g. 192.1.1.1), a range of IP values (e.g. 192.1.1.1 – 192.1.2.1) or `*` (all values);
- t is the protocol type that can be `TCP`, `UDP` or `*` (all values);
- *p_src* and *p_dst* are respectively the ranges of the source and destination port numbers ;
- a is the policy action that the firewall must carry out if the received traffic matches with the current policy and that can assume either `accept` or `deny` as value.

The set of relations $\mathfrak{R}$ that can exist between network fields of pi_i and pi_j depends on their value types (e.g. f is an integer, while *ip_src* is an IP address) and can be summarized as follows:

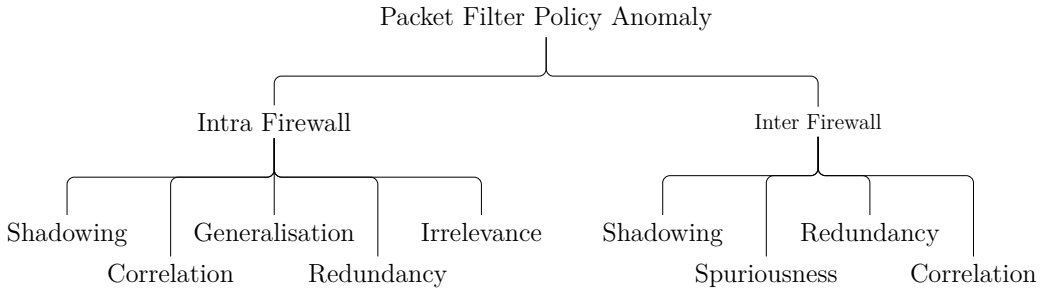


Fig. 9.1 Al-Shaer's classification of Packet Filter policy anomalies.

- f_i and r_i can be equal to or can dominate the relative field of pi_j (e.g. $f_i \succ f_j$, if f_i is equal to 6, while f_j is equal to 3);
- ip_src, ip_dst, p_src and p_dst can be equal, disjointed or can dominate the relative fields of pi_j;
- finally, a_i and t_i can be equal or disjointed to t_j and a_j.

Concerning the definition of detection rules, we can leverage the anomaly classification proposed in [8]. As show in Chapter 3, the authors have paid attention to two main types of conflicts between FPs, that are the Intra- and Inter-firewall anomalies (Fig. 9.1):

- *Intra-Firewall Shadowing anomaly*

  $f_i = f_j \wedge r_j \succ r_i \wedge ip\_src_i \succeq ip\_src_j \wedge t_i \succeq t_j \wedge ip\_dst_i \succeq ip\_dst_j \wedge$
  $p\_src_i \succeq p\_src_j \wedge p\_dst_i \succeq p\_dst_j \wedge a_i \neq a_j \Rightarrow$ Intra-Shadowing(pi_i,pi_j)

- *Intra-Firewall Correlation anomaly*

  $f_i = f_j \wedge ip\_src_i \not\preceq ip\_src_j \wedge t_i \not\preceq t_j \wedge ip\_dst_i \not\preceq ip\_dst_j \wedge$
  $p\_src_i \not\preceq p\_src_j \wedge p\_dst_i \not\preceq p\_dst_j \wedge a_i \neq a_j \Rightarrow$ Intra-Correlation(pi_i,pi_j)

- *Intra-Firewall Generalization anomaly*

  $f_i = f_j \wedge r_i \succ r_j \wedge ip\_src_i \succ ip\_src_j \wedge t_i \succ t_j \wedge ip\_dst_i \succ ip\_dst_j \wedge$
  $p\_src_i \succ p\_src_j \wedge p\_dst_i \succ p\_dst_j \wedge a_i \neq a_j \Rightarrow$ Intra-Generalization(pi_i,pi_j)

- *Intra-Firewall Redundancy anomaly*

  $f_i = f_j \wedge r_j \succ r_i \wedge ip\_src_i \succeq ip\_src_j \wedge t_i \succeq t_j \wedge ip\_dst_i \succeq ip\_dst_j \wedge$
  $p\_src_i \succeq p\_src_j \wedge p\_dst_i \succeq p\_dst_j \wedge a_i = a_j \Rightarrow$ Intra-Redundancy(pi_i,pi_j)

- *Inter-Firewall Shadowing anomaly*

  $f_i \succ f_j \wedge ip\_src_i \succeq ip\_src_j \wedge t_i \succeq t_j \wedge ip\_dst_i \succeq ip\_dst_j \wedge p\_src_i \succeq p\_src_j \wedge$
  $p\_dst_i \succeq p\_dst_j \wedge a_i =$ accept $\wedge\, a_j =$ deny $\Rightarrow$ Inter-Shadowing(pi_i,pi_j)

  $f_i \succ f_j \wedge ip\_src_j \succ ip\_src_i \wedge t_j \succ t_i \wedge ip\_dst_j \succ ip\_dst_i \wedge p\_src_j \succ p\_src_i \wedge$
  $p\_dst_j \succ p\_dst_i \wedge a_i =$ accept $\wedge\, a_j =$ deny $\Rightarrow$ Inter-Shadowing(pi_i,pi_j)

- *Inter-Firewall Spuriousness anomaly*

  $f_i \succ f_j \wedge ip\_src_j \succeq ip\_src_i \wedge t_j \succeq t_i \wedge ip\_dst_j \succeq ip\_dst_i \wedge p\_src_j \succeq p\_src_i \wedge$
  $p\_dst_j \succeq p\_dst_i \wedge a_j =$ accept $\wedge\, a_i =$ deny $\Rightarrow$ Inter-Spuriousness(pi_i,pi_j)

  $f_i \succ f_j \wedge ip\_src_i \succ ip\_src_j \wedge t_i \succ t_j \wedge ip\_dst_i \succ ip\_dst_j \wedge p\_src_i \succ p\_src_j \wedge$
  $p\_dst_i \succ p\_dst_j \wedge a_j =$ accept $\wedge\, a_i =$ deny $\Rightarrow$ Inter-Spuriousness(pi_i,pi_j)

- *Inter-Firewall Redundancy anomaly*

  $f_i \succ f_j \wedge ip\_src_i \succeq ip\_src_j \wedge ip\_dst_i \succeq ip\_dst_j \wedge p\_src_i \succeq p\_src_j \wedge$
  $p\_dst_i \succeq p\_dst_j \wedge t_i \succeq t_j \wedge a_i = a_j =$ deny $\Rightarrow$ Inter-Redundancy(pi_i,pi_j)

- *Inter-Firewall Correlation anomaly*

  $f_i \succ f_j \wedge ip\_src_i \not\preceq ip\_src_j \wedge t_i \not\preceq t_j \wedge ip\_dst_i \not\preceq ip\_dst_j \wedge$
  $p\_src_i \not\preceq p\_src_j \wedge p\_dst_i \not\preceq p\_dst_j \Rightarrow$ Inter-Correlation(pi_i,pi_j)

9.2 Communication Protection Policy

The second use case we consider to apply the UMPA model is the communication protection policy. In this scenario, we review the concepts and the model presented in Chapter 5 in order to map the Inter-technology analysis of this type of policies in the UMPA model.

Thus the *pi* needed to model a communication protection policy in this unified model takes the following structure:

$$i = (r, s, d, t, C, S, G)$$

where:

- r is an incremental rule identifier specific for the priority (the lower the number the higher the priority);
- s and d symbolize the source and destination nodes and their possible values could be either the network node id (`Node`) or a its IP address (`IP`), port number (`Port`) or URI (`URI`);
- t specifies the adopted technology, that could be `IPsec`, `TLS`, `SSH`, `WS-Security` or `NULL`;
- C characterizes the policy actions and are three security coefficients values that denote a required security level for a specific property (header integrity c^{hi}, payload integrity c^{pi} and confidentiality c^{c});
- S is a tuple of network fields (i.e. Selector), used to identify the traffic that need to be protected;
- G is an order list of the gateways involved in the communication;

The values s, d, S, G are network fields, while the values t, C are actions. We also consider as extra network fields the function described in Chapter 5:

- $\mathcal{N}(i_1)$ that returns the node where the PI is actually deployed;
- $\mathcal{T}(e)$ that returns the set of technologies supported by the node e;

- $C_{max}(i_1)$ that returns the set of maximum enforceable coefficients by PI i_1;
- $C_{min}(i_1)$ that returns the minimum acceptable coefficients for the channel defined by PI i_1;

The CPP detection rules in the UMPA model are:

- *Internal loop*

$$s_1 \not\perp d_1 \Rightarrow \mathcal{A}_{il}(i_1)$$

- *Out of place*

$$\mathcal{N}(i_1) \perp s_1 \Rightarrow \mathcal{A}_{op}(i_1)$$

- *Non-enforceability*

$$C_1 \succ \mathcal{C}_{max}(i_1) \vee \mathcal{T}(s_1) \not\succ t_1 \vee t_1 \notin \mathcal{T}(d_1) \Rightarrow \mathcal{A}_{ne}(i_1)$$

- *Inadequacy*

$$C_1 \prec \mathcal{C}_{min}(i_1) \Rightarrow \mathcal{A}_{in}(i_1)$$

- *Shadowing*

$$r_1 \prec r_2 \wedge t_1 = t_2 \wedge s_1 \succeq s_2 \wedge d_1 \preceq d_2$$
$$\wedge S_1 \succeq S_2 \wedge C_1 \perp C_2 \wedge G_1 = G_2 \Rightarrow \mathcal{A}_{sh}(i_1, i_2)$$

- *Redundancy*

$$t_1 = t_2 \wedge s_1 \succeq s_2 \wedge d_1 \preceq d_2 \wedge$$
$$\wedge S_1 \succeq S_2 \wedge C_1 \succeq C_2 \wedge G_1 = G_2 \Rightarrow \mathcal{A}_{re}(i_1, i_2)$$

- *Exception*

$$r_1 \prec r_2 \wedge t_1 = t_2 \wedge s_1 \prec s_2 \wedge d_1 \prec d_2 \wedge$$
$$\wedge d_1 \prec d_2 \wedge S_1 \succ S_2 \wedge C_1 \perp C_2 \wedge G_1 = G_2 \Rightarrow \mathcal{A}_{ex}(i_1, i_2)$$

- *Correlation*

$$s_1 \not\preceq s_2 \wedge d_1 \not\preceq d_2 \wedge t_1 = t_2 \wedge G_1 = G_2 \wedge S_1 = S_2 \wedge$$
$$\wedge \neg \mathcal{C}_{sh}(i_1, i_2) \wedge \neg \mathcal{C}_{ex}(i_1, i_2) \wedge \neg \mathcal{C}_{re}(i_1, i_2) \Rightarrow \mathcal{A}_{co}(i_1, i_2)$$

  where $\mathcal{C}_{sh}(i_1, i_2)$, $\mathcal{C}_{ex}(i_1, i_2)$, $\mathcal{C}_{re}(i_1, i_2)$, are respectively the set of condition of Shadowing, Exception and Redundancy anomaly

- *Inclusion*

$$s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge t_1 \succeq t_2 \wedge C_1 \succeq C_2$$
$$\wedge S_1 \succeq S_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2 \Rightarrow \mathcal{A}_{in}(i_1, i_2)$$

- *Affinity*

$$(s_1 \not\preceq s_2 \wedge d_1 \not\preceq d_2 \wedge t_1 \not\preceq t_2) \wedge$$
$$\wedge \neg \mathcal{C}_{in}(i_1, i_2) \wedge \neg \mathcal{C}_{in}(i_2, i_1) \Rightarrow \mathcal{A}_{af}(i_1, i_2)$$

- *Contradiction*

$$s_1 \not\preceq s_2 \wedge d_1 \not\preceq d_2 \wedge t_1 \perp t_2 \Rightarrow \mathcal{A}_{co}(i_1, i_2)$$

9.3 Traffic Flow Policy

Service Function Chaining (SFC) [103] defines an ordered set of network functions (e.g. NAT, load balancer, web cache) that processes the incoming traffic. Here, forwarding policies can be used to configure the traffic flows that must be processed by a given chain.

In literature, most of the existing works focused on the OpenFlow protocol, which has been a successful SFC implementation. The detection of errors among OpenFlow rules deployed into the OpenFlow switches has been addressed in depth, while the correct forwarding policy definition at the SDN controller layer was overlooked.

In order to validate that the proposed model is able to support administrator-defined policy anomalies, we propose a new definition of *PI* structure and

detection rules to detect forwarding policy anomalies at controller layer, in the flow-based domain. The proposed *PI* is structured as follows:

$$pi = (ip\_src,\ ip\_dst,\ t,\ p\_src,\ p\_dst,\ C)$$

- *ip_src* and *ip_dst* symbolize the source and destination IP address of the PI;
- t is the protocol type and can assume `TCP` or `UDP` as value;
- *p_src* and *p_dst* specify the ranges of the source and destination port numbers;
- C is a parameter of the policy action that specifies the ordered set of network functions towards traffic will be forwarded.

In addition, the possible relation $\mathfrak{R}$ between PIs (pi_i and pi_j) could be similar to the filtering policy case:

- ip_src_i, ip_dst_i, p_src_i and p_dst_i can be equal, disjoint or can dominate the corresponding fields of pi_j;
- t_i can be equal or disjoint to t_j;
- C_i can be equal, disjointed, correlated or dominate to C_j.

For the sake of simplicity, the various network fields of a PI, apart from C, are grouped under a single symbol, N. Hence the set of network fields N_i of pi_i will have a relation $\mathfrak{R}$ with the N_j of the pi_j, if certain conditions are verified. We recall that those relations can be *equivalence*, *dominance*, *disjointness* or *correlation*.

Finally, we also propose an example of possible *PI* anomalies between SFCPs, which are represented by the following detection rules:

- *Intra-PI Incorrect anomaly* arises when the source and destination of a traffic flow correspond. This means that the *ip_src* and *ip_dst* fields of pi_i are equal and, hence, a forwarding loop is generated in the network

$$ip\_src_i = ip\_dst_i \Rightarrow \text{Incorrect}(pi_i)$$

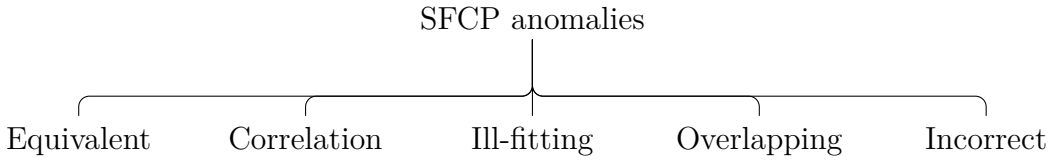


Fig. 9.2 An example of anomalies classification of Flow-based policies.

- *Inter-PI Equivalent anomaly* occurs if all network fields of pi_i and pi_j are equal

$$N_i = N_j \wedge C_i = C_j \Rightarrow \text{Equivalent}(pi_i,pi_j)$$

- *Inter-PI Ill-fitting Anomaly* occurs if N_i dominates N_j and C_j dominates C_i. N_i must be redefined to be disjointed to N_j

$$N_i \succ N_j \wedge C_j \succ C_i \Rightarrow \text{Ill-fitting}(pi_i,pi_j)$$

- *Inter-PI Overlapping Anomaly* occurs when N_i is correlated to N_j and C_i dominates C_j. In this case, pi_i and pi_j must be split into a set of pi that will be disjointed from each other

$$N_i \sim N_j \wedge C_i \succ C_j \Rightarrow \text{Overlapping}(pi_i,pi_j)$$

- *Inter-PI Inter-PI Correlation anomaly* occurs if all network fields of pi_i and pi_j are equal, except C_i and C_j that are correlated. Hence we will create an ambiguity in traffic forwarding because we have to indicate the correct chain towards forward traffic:

$$N_i = N_j \wedge C_i \sim C_j \Rightarrow \text{Correlated}(pi_i,pi_j)$$

9.4 Inter-domain Analysis

Until now, we have shown how the UMPA model can map existing and novel domain, by applying Intra-domain analysis as the literature does. In fact, the detection of policy anomalies defined among policies of different domains

(i.e., Inter-domain anomaly) was generally overlooked by the existing works on policy analysis. An Inter-domain analysis would allow to check a greater and more comprehensive set of anomalies than the set that the state of the art approaches are able to detect. By means of this novel type of analysis, thus, administrators will achieve a higher flexibility in the definition of network errors, events and redundancies.

Certainly it is very difficult to have an exhaustive and objective criteria to establish when a network scenario is a misconfiguration or if it is a weakened scenario. In fact, the administrator is the only one who can decide which anomaly is really an undesired situation. For this reason, we show how the UMPA model can support an Inter-domain analysis by providing some examples of this novel type of anomalies. In particular, we use the following examples because they may be reasonably an anomalies in many network scenarios.

Having presented the Filtering and Communication Protection Policies, a first example can be the *Pair-PI Filtered Anomaly*. In particular such anomaly identifies when a certain traffic flow, belonging to a secured communication, is discarded by a firewall because a rule is installed to deny ($a_j = \texttt{deny}$) such flow. This means that the secure communication (implemented by the CPP pi_i) is interrupted by the FP pi_j installed in the firewall. Hence, in the proposed model, the Pair-PI Filtered Anomaly can be expressed in this way:

$$s_i \not\perp ip\_src_j \wedge s_i \not\perp p\_src_j \wedge d_i \not\perp ip\_dst_j \wedge$$
$$d_i \not\perp p\_dst_j \wedge a_j = \texttt{deny} \Rightarrow \text{Filtered}(pi_i, pi_j)$$

The last example is the Inter-PI Interruption Anomaly that arises between the FP and SFC domains: let us consider a traffic flow (K) that should be processed by a service chain that contains a firewall ($C_i \succ \{f_j\}$), but a filtering policy was defined so that the flow K must be dropped by the firewall.

This means that the flow K will not traverse the entire service chain:

$$ip\_src_i = ip\_src_j \wedge ip\_dst_i = ip\_dst_j \wedge t_i = t_j \wedge p\_src_i = p\_src_j \wedge$$
$$p\_dst_i = p\_dst_j \wedge a_j = \{\texttt{deny}\} \wedge C_i \succ \{f_j\} \wedge \Rightarrow \text{Interruption}(pi_i, pi_j)$$

Summary on Network Security Policy Analysis

Chapter 10

Conclusion and Future Works

In this last chapter of the thesis, we recap our contribution in the state of arts and the possible future works to continue extend this topic.

10.1 Conclusion

This dissertation has proposed two novel types of anomaly analysis of network security policy: *Inter-technology and Inter-domains.*

Policy Anomaly Analysis is designed to identify potential errors, conflict and redundancy among policy rules. In literature, several works and techniques have been proposed to identify anomalies, however the research is mainly concentrated on Intra- and Inter-policy analysis. The Intra-Policy analysis identifies any anomaly among the rules of a single policy, while the Inter-Policy analysis identifies anomalies among the rules of a set of interconnected policies. However, the complexity of real systems is not self-contained, as each network security control may affect the behavior of other controls in the same network.

For this reason, we propose to extend the policy anomaly analysis with two new types of analysis: Inter-technology and Inter-domain.

The *Inter-technology analysis* identifies any anomaly among a set of policies that enforce different security communication technologies (e.g. IPsec, TLS, SSH). For instance, when an IPsec tunnel encapsulates other TLS tunnels, the external tunnel is a redundant level of protection.

The *Inter-domain analysis* identifies anomalies in a set of policies of different security policy domains. This is the case of a firewall that blocks some encrypted communication channels created by a VPN functionalities, which generates an Inter-domain anomaly between the filtering domain (i.e. firewall) and the communication-protection one (i.e. VPN).

In order to validate the effectiveness of the Inter-technology analysis, we have applied it on communication protection policy.

Communication protection policies specify how to protect the network communications. Their correct deployment is crucial in several areas, such as protection of intellectual properties, and confidentiality of financial or corporate data (like credit card numbers).

The presented work on the analysis on communication protection policy extends the current literature adding new list of nineteen anomalies that can appear during the implementation or design of communication protection policies. The proposed anomalies are classified in two taxonomies: an effect-base taxonomy and an information-specific taxonomy. Through an empirical assessment we have proved the practical significance of detecting this new class of anomalies.

Furthermore, during this analysis we have introduced a formal model, based on first-order logic rules that analyses the network topology and the security controls at each node to identify the detected anomalies and suggest the strategies to resolve them.

The opposed model, implement the Inter-technology analysis allowing the detection of a number of anomalies arising from the interactions between various protocols (e.g. TLS and SSH), security properties and communication scenarios such as end-to-end connections, VPNs and remote access communications (see RFC-3457 [104]). We took into account both communication end-points (i.e. source and destination), but also tunnel terminators/gateways for a more accurate detection. To the best of our knowledge this is the first work that detects and classifies communication Inter-technology policy anomalies. Our approach internally represents every network device as a tree containing various "entities", able to establish or terminate secure communications, which live at different ISO/OSI levels. Our hierarchical view of networks and network nodes

improves on existing works, which often only rely on a flat IP address-based representation of an IT infrastructure.

In addition, we have proposed a graphical view of the detected anomalies with a simple and intuitive representation, thus facilitating their identification. This representation is based on multi-graph theory, and it is an equivalent model with respect to the FOL model.

Moreover, we have performed the Inter-domain analysis defining a unified model for policy analysis.

This unified analysis model is composed by two parts: the policies implementation and anomalies detection rules.

- a Policy Implementation, in our model, represents the policy structure and it is very flexible and customizable. A Policy Implementation identifies the conditions and events that administrators want to manage and the policy actions that describe the way those events are managed.
- a *detection rule* is a set of conditions that distinguish the possible anomalies among Policy Implementations. The detection rules are based on First Order Logic (FOL) and are expressed using Horn clauses.

The generality of the solution was also validated by applying the model to three case studies of different domains: communication protection, data filtering and service function chaining.

10.2 Future Works

In the near future, we plan to extend the expressivity and capabilities of our unified model adding support for a smart and optimized solution of the conflicts. This will increase the performance of the network without altering its semantics.

In the medium term, we also plan to add in our model other policy analysis techniques like policy reachability and policy reconciliation. Unfortunately, this will require a deeper knowledge of the network state to ensure that the best policies are chosen and to reconcile policy conflicts.

Finally for a next future, we also aim to integrate the policy anomaly analysis module in the Network Functions Virtualization (NFV) and Software-Defined Networks (SDN).

Thanks to the introduction of NFV and SDN, the concept of *Service Function Chain (SFC)* has been defined in literature [103]. A service function chain defines an ordered set of abstract service functions and ordering constraints that must be applied to packets and/or flows selected (as a result of classification). An example of an abstract service function is "a firewall".

The challenges of service function chain are the same of security controls, like the complexity of configuring such services or putting them in the correct order [105].

As it is already envisioned by the I2NSF [106] working group in ETSI, the configuration of single function is not enough to enforce security in the whole network, because administrators have to consider also the interactions between different security functions in a distributed environment for making the network secure.

For these reasons, our goals are: (i) extend the capability of the UMPA model in order to support the technology that implements the SFC; (ii) integrate the policy analysis module in an open-source NFV implementation. This would provide the NFV orchestrator with additional information, increasing both the security and the efficiency of the deployment.

References

[1] Avishai Wool. Trends in firewall configuration errors: Measuring the holes in swiss cheese. *IEEE Internet Computing*, 14(4):58–65, July 2010.

[2] Verizon. 2014 Data Breach Investigations Report. Technical report, Verizon, May 2014. http://www.nu.nl/files/Verizon.pdf.

[3] Verizon. 2015 Data Breach Investigations Report. Technical report, Verizon, May 2015. https://iapp.org/media/pdf/resource_center/Verizon_data-breach-investigation-report-2015.pdf.

[4] Verizon. 2016 Data Breach Investigations Report. Technical report, Verizon, May 2016. http://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf.

[5] Center for Strategic and International Studies. Securing cyberspace for the 44th presidency. Technical report, December 2008. https://www.nitrd.gov/cybersecurity/documents/081208_securingcyberspace_44.pdf.

[6] John Strassner. How policy empowers business-driven device management. In *Proc. of the 3rd International Workshop on Policies for Distributed Systems and Networks*, pages 214–217, Monterey, California, USA, June 2002. IEEE.

[7] Zhi Fu, Shyhtsun Felix Wu, He Huang, Kung Loh, Fengmin Gong, Ilia Baldine, and Chong Xu. IPsec/VPN security policy: Correctness, conflict detection, and resolution. In *International Workshop on Policies for Distributed Systems and Networks*, POLICY'01, pages 39–56, Bristol, United Kingdom, January 2001. Springer-Verlag.

[8] Ehab Al-Shaer, Hazem Hamed, Raouf Boutaba, and Masum Hasan. Conflict classification and analysis of distributed firewall policies. *IEEE Journal on Selected Areas in Communications*, 23(10):2069–2084, October 2005.

[9] Nayan Basumatary and Shyamanta M Hazarika. Model checking a firewall for anomalies. In *Proc. of the 1st International Conference on Emerging Trends and Applications in Computer Science*, pages 92–96, Shillong, India, September 2013. IEEE.

[10] Cisco Systems. User Guide for Cisco Security Manager 4.3. Technical report, December 2015. http://www.cisco.com/c/en/us/td/docs/security/security_management/cisco_security_manager/security_manager/410/user/guide/CSMUserGuide.pdf.

[11] David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 184–195, Oakland, CA, USA, April 1987. IEEE.

[12] D.C. Robinson and Morris S. Sloman. Domains: a new approach to distributed system management. In *Proc. of the IEEE Workshop on the Future Trends of Distributed Computing Systems*, pages 154–163, Hong Kong, Cina, September 1988. IEEE.

[13] Andrea Westerinen, John Schnizlein, John Strassner, Mark Scherling, Bob Quinn, Shai Herzog, A Huynh, Mark Carlson, Jay Perry, and Steve Waldbusser. Terminology for Policy-Based Management. RFC 3198 (Informational), November 2001. http://www.rfc-editor.org/rfc/rfc3198.txt.

[14] Bob Moore, Ed Ellesson, John Strassner, and Andrea Westerinen. Policy Core Information Model – Version 1 Specification. RFC 3060 (Proposed Standard), February 2001. http://www.rfc-editor.org/rfc/rfc3060.txt.Updated by RFC 3460.

[15] Raouf Boutaba and Issam Aib. Policy-based management: A historical perspective. *Journal of Network and Systems Management*, 15(4):447–480, November 2007.

[16] Ehab S. Al-Shaer and Hazem H. Hamed. Firewall policy advisor for anomaly discovery and rule editing. In *Proc. of theIEEE Eighth International Symposium on Integrated Network Management.*, pages 17–30, Boston, Massachusetts, USA, March 2003. IEEE.

[17] Joaquin Garcia-Alfaro, Frédéric Cuppens, Nora Boulahia, Salvador Martinez, and Jordi Cabot. Management of stateful firewall misconfiguration. *Computers & Security*, 39(A):64–85, November 2013.

[18] Alex X. Liu and Mohamed G. Gouda. A model of stateful firewalls and its properties. In *Proc. of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, volume 00, pages 128–137, Los Alamitos, CA, USA, November 2005. IEEE.

[19] Cataldo Basile and Antonio Lioy. Analysis of Application-Layer Filtering Policies With Application to HTTP. *IEEE/ACM Transactions on Networking*, 23(1):28–41, December 2015.

[20] Elisa Bertino, Elena Ferrari, and Andrea Perego. A general framework for web content filtering. *World Wide Web*, 13(3):215–249, October 2010.

[21] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, June 2015.

[22] Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy. Network innovation using OpenFlow: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):493–512, February 2014.

[23] European Commission. Directive 95/46/ec of the european parliament and of the council on the adequacy of the protection provided by the safe harbour privacy principles and related frequently asked questions issued by the us department of commerce. Technical report, European Parliament, July 2000. http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32000D0520&from=EN.

[24] Sheila Frankel and Suresh Krishnan. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, RFC Editor, February 2011. http://www.rfc-editor.org/rfc/rfc6071.txt.

[25] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, RFC Editor, August 2008. http://www.rfc-editor.org/rfc/rfc5246.txt.

[26] Tatu Ylonen and Chris Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, RFC Editor, January 2006. http://www.rfc-editor.org/rfc/rfc4251.txt.

[27] Alan O. Freier, Philip Kariton, and Paul C. Kocher. WS-Security 2004. Oasis standard specification, OASIS, 2006. https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[28] Loki Radoslav. *SSH File Transfer Protocol*. PON PRESS, 2012.

[29] George Tsirtsis and Pyda Srisuresh. Network Address Translation - Protocol Translation (NAT-PT). RFC 2766 (Historic), February 2000. Obsoleted by RFC 4966, updated by RFC 3152.

[30] Jiang Qian, Susan Hinrichs, and Klara Nahrstedt. ACLA: A Framework for Access Control List (ACL) Analysis and Optimization. In *Proc. of the 5th Joint Working Conference on Communications and Multimedia Security (CMS01)*, pages 197–211, Darmstadt, Germany, May 2001. Springer Nature.

[31] Norbert Lehmann, Reinhard Schwarz, and Jörg Keller. FIRECROCODILE: A Checker for Static Firewall Configurations. In *Proc. of the International Conference on Security & Management (SAM06)*, pages 193–199, Las Vegas, Nevada, USA, June 2006. CSREA Press.

[32] Lihua Yuan, Hao Chen, Jianning Mai, and Chen-nee Chuah. FIREMAN: A Toolkit for FIREwall Modeling and ANalysis. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 199–213, Oakland, California, USA, May 2006. IEEE.

[33] Alan Jeffrey and Taghrid Samak. Model Checking Firewall Policy Configurations. In *Proc. of the IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY2009)*, pages 60–67, London, United Kingdom, July 2009. IEEE.

[34] Simone Ferraresi, Stefano Pesic, Livia Trazza, and Andrea Baiocchi. Automatic conflict analysis and resolution of traffic filtering policy for firewall and security gateway. In *Proc. of the IEEE International Conference on Communications (ICC'07)*, pages 1304–1310, Glasgow, Scotland, United Kingdom, June 2007. IEEE.

[35] Korosh Golnabi, Richard K Min, Latifur Khan, and Ehab Al-Shaer. Analysis of firewall policy rules using data mining techniques. In *Proc. of the10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, pages 305–315, Vancouver, Canada, April 2006. IEEE.

[36] Alex X. Liu and Mohamed G. Gouda. Complete Redundancy Detection in Firewalls. In *Proc. of the 19th Working Conference on Data and Applications Security (IFIP WG 11.3)*, pages 193–206, Storrs, Connecticut, USA, August 2005. Springer.

[37] Alex Liu and Mohamed Gouda. Structured firewall design. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(4):1106–1120, March 2007.

[38] Frédéric Cuppens, Nora Cuppens-Boulahia, and Joaquin Garcia-Alfaro. Detection and Removal of Firewall Misconfiguration. In *Proc. of the 2005 IASTED International Conference on Communication, Network and Information Security*, pages 154–162, Phoenix, Arizona, USA, November 2005. IASTED.

[39] Frédéric Cuppens, Nora Cuppens-Boulahia, and Joaquin Garcia-Alfaro. Detection of network security component misconfiguration by rewriting and correlation. In *Proc. of the 1th joint conference on security in network architectures (SAR) and security of information systems (SSI)*, pages 6–9, Seignosse, Landes, France, June 2006.

[40] Joaquin Garcia-Alfaro, Nora Cuppens-Boulahia, and Frédéric Cuppens. Complete analysis of configuration rules to guarantee reliable network security policies. *International Journal of Information Security*, 7(2):103–122, April 2007.

[41] Joaquin Garcia-Alfaro, Frédéric Cuppens, Nora Boulahia, and Preda Stere. MIRAGE: a management tool for the analysis and deployment of network

security policies. In *Proc. of the 3rd International Workshop (SETOP 2010)*, pages 203–215, Athens, Greece, September 2011. Springer-Verlag.

[42] Tarek Abbes, Adel Bouhoula, and Michaël Rusinowitch. An inference system for detecting firewall filtering rules anomalies. In *Proc. of the ACM symposium on Applied computing (SAC08)*, pages 2122–2128, Fortaleza, Brazil, March 2008. ACM.

[43] Amina Saadaoui, Nihel Ben Youssef Ben Souayeh, and Adel Bouhoula. Formal approach for managing firewall misconfigurations. In *Proc. of the IEEE 8th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–10, Marrakech, Morocco, May 2014. IEEE.

[44] Muhammad Abedin, Syeda Nessa, Latifur Khan, and Bhavani M. Thuraisingham. Detection and resolution of anomalies in firewall policy rules. In *Proc. of the 20th Annual IFIP WG Working Conference on Data and Applications Security,*, pages 15–29, Sophia Antipolis, France, July 2006. Springer-Verlag.

[45] Cataldo Basile, Alberto Cappadonia, and Antonio Lioy. Network-Level Access Control Policy Analysis and Transformation. *IEEE/ACM Transactions on Networking*, 20(4):985–998, August 2012.

[46] Cataldo Basile, Alberto Cappadonia, and Antonio Lioy. Geometric interpretation of policy specification. In *Proc. of the IEEE Workshop on Policies for Distributed Systems and Networks (POLICY 2008)*, pages 78–81, Palisades, New York, USA, June 2008. IEEE.

[47] Hongxin Hu, Gail-Joon Ahn, and Ketan Kulkarni. FAME: a firewall anomaly management environment. In *Proc. of the 3rd ACM workshop on Assurable and usable security configuration (SafeConfig10)*, pages 17–26, Chicago, USA, October 2010. ACM.

[48] Hongxin Hu, Gail Joon Ahn, and Ketan Kulkarni. Detecting and resolving firewall policy anomalies. *IEEE Trans. Dependable Sec. Comput.*, 9(3):318–331, May 2012.

[49] Wadie Krombi, Mohammed Erradi, and Ahmed Khoumsi. Automata-Based Approach to Design and Analyze Security Policies. In *Proc. of the 12th International Conference on Privacy, Security and Trust (PST2014)*, pages 306–313, Toronto, Canada, July 2014. IEEE.

[50] Ahmed Khoumsi, Wadie Krombi, and Mohammed Erradi. A formal approach to verify completeness and detect anomalies in firewall security policies. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8930:221–236, July 2015.

[51] Frédéric Cuppens. Handling Stateful Firewall Anomalies. In *Proc. of the Information Security and Privacy Conference (SEC2012)*, pages 174–186, Heraklion, Greece, June 2012. Springer, Berlin, Heidelberg.

[52] Cataldo Basile and Antonio Lioy. Analysis of Application-Layer Filtering Policies With Application to HTTP. *IEEE/ACM Transactions on Networking*, PP(99):1–1, December 2013.

[53] Ehab Al-Shaer and Saeed Al-Haj. FlowChecker: configuration analysis and verification of federated openflow infrastructures. In *Proc. of the 3rd ACM workshop on Assurable and usable security configuration (SafeConfig10)*, pages 37–44, Chicago, USA, October 2010. ACM.

[54] Marco Canini, Daniele Venzano, Peter Perešíni, Dejan Kostić, and Jennifer Rexford. A nice way to test openflow applications. In *Proc. of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI12)*, pages 10–10, San Jose, CA, April 2012. USENIX Association.

[55] Bruno Lopes Alcantara Batista, Gustavo Augusto Lima de Campos, and Marcial P Fernandez. Flow-based conflict detection in openflow networks using first-order logic. In *Proc. of the IEEE Symposium on Computers and Communication (ISCC2014)*, pages 1–6, Funchal, Portugal, June 2014. IEEE.

[56] John Zao. Semantic model for IPsec policy interaction. Technical report, Internet Draft, March 2000. https://tools.ietf.org/html/draft-zao-policy-semantics-00.

[57] Hazem Hamed, Ehab Al-Shaer, and Will Marrero. Modeling and Verification of IPSec and VPN Security Policies. In *Proc. of the 13th IEEE Int. Conference on Network Protocols*, pages 259–278, Boston, MA, November 2005. IEEE.

[58] Ehab Al-Shaer and Hazem Hamed. Taxonomy of conflicts in network security policies. *IEEE Communications Magazine*, 44(3):134–141, March 2006.

[59] Salman Niksefat and Masoud Sabaei. Efficient Algorithms for Dynamic Detection and Resolution of IPSec/VPN Security Policy Conflicts. In *Proc. of the 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 737–744, Perth, Western Australia, April 2010. IEEE.

[60] Zhitang Li, Xue Cui, and Lin Chen. Analysis And Classification of IPSec Security Policy Conflicts. In *Proc. of the Japan-China Joint Workshop on Frontier of Computer Science and Technology (FCST06)*, pages 83–88, Fukushimna, Japan, November 2006. IEEE.

[61] Alex X. Liu and Mohamed G. Gouda. Firewall Policy Queries. *IEEE Transactions on Parallel and Distributed Systems*, 20(6):766–777, June 2009.

[62] Adel El-Atawy, Taghrid Samak, Zein Wali, Ehab Al-Shaer, Frank Lin, Christopher Pham, and Sheng Li. An Automated Framework for Validating Firewall Policy Enforcement. In *Proc. of the 8th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY07)*, pages 151–160, Bologna, Italy, June 2007. IEEE.

[63] Ehab Al-Shaer, Adel El-Atawy, and Taghrid Samak. Automated pseudo-live testing of firewall configuration enforcement. *IEEE Journal on Selected Areas in Communications*, 27(3):302–314, April 2009.

[64] Achim D. Brucker, Lukas Brügger, and Burkhart Wolff. hol-TestGen/fw. In *Proc. of the 10th International Colloquium*, pages 112–121, Shanghai, China, September 2013. Springer.

[65] Alain Mayer, Avishai Wool, and Elisha Ziskind. Fang: a firewall analysis engine. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 177–187, Berkeley, CA, May 2000. IEEE.

[66] Avishai Wool. Architecting the Lumeta Firewall Analyzer. In *Proc. of the 10th USENIX Security Symposium*, pages 85–97, Washington, DC, August 2001. USENIX.

[67] Alain Mayer, Avishai Wool, and Elisha Ziskind. Offline firewall analysis. *International Journal of Information Security*, 5(3):125–144, July 2006.

[68] Robert Marmorstein and Phil Kearns. A Tool for Automated iptables Firewall Analysis. In *Proc. of the USENIX Annual Technical Conference (ATEC05)*, pages 71–81, Anaheim, CA, April 2005. USENIX.

[69] Robert Marmorstein and Phil Kearns. An open source solution for testing NAT'd and nested iptables firewalls. In *Proc. of the 19th Large Installation Systems Administration Conference (LISA05)*, pages 103–112, San Diego, CA, December 2005. USENIX.

[70] Robert Marmorstein and Phil Kearns. Debugging a firewall policy with policy mapping. *;login: the USENIX Association newsletter*, 32(1):44–51, February 2007.

[71] Ricardo Oliveira, Sihyung Lee, and Hyong Kim. Automatic detection of firewall misconfigurations using firewall and network routing policies. In *Proc. of the Workshop on proactive failure avoidance, recovery and maintenance (PFARM)*, Lisbon, Portugal, June 2009. IEEE.

[72] Pasi Eronen and Jukka Zitting. An expert system for analyzing firewall rules. In *Proc. of the 6th Nordic Workshop on Secure IT Systems (NordSec2001)*, pages 100–107, Lyngby, Denmark, June 2001.

[73] Scott Hazelhurst. Algorithms for Verifying Firewall and Router Access Lists. In *Proc. of the 46th Midwest Symposium on Circuits and Systems*, pages 512–515, Cairo, Egypt, December 2003. IEEE.

[74] Geoffrey G. Xie, Jibin Zhan, David .A Maltz, Hui Zhang, Albert Greenberg, Gisli Hjalmtysson, and Jennifer Rexford. On static reachability analysis of IP networks. In *Proc. of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM2005)*, pages 2170–2183, Miami, FL, March 2005. IEEE.

[75] Sruthi Bandhakavi, Sandeep Bhatt, Cat Okita, and Prasad Rao. Analyzing end-to-end network reachability. In *Proc. of the IFIP/IEEE Int. Symposium on Integrated Network Management (IM09)*, pages 585–590, Long Island, NY, June 2009. IEEE.

[76] Petr Matoušek, Jaroslav Ráb, Ondřej Ryšavý, and Miroslav Švéda. A formal model for network-wide security analysis. In *Proc. of the 15th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS2008)*, pages 171–181, Belfast, UK, April 2008. IEEE.

[77] Mikkel Christiansen and Emmanuel Fleury. An Interval Decision Diagram Based Firewall. In *Proc. of the 3rd IEEE International Conference on Networking (ICN '04)*, Point-à-Pitre, Guadeloupe, April 2004. IEEE.

[78] A. Khakpour and Alex Liu. Quarnet: A Tool for Quantifying Static Network Reachability. *IEEE/ACM Transactions on Networking*, 21(2):551 – 565, February 2009.

[79] A. Khakpour and Alex Liu. Quantifying and querying network reachability. pages 817–826, February 2010.

[80] Alex Liu and A. Khakpour. Quantifying and verifying reachability for access controlled networks. *IEEE/ACM Transactions on Networking*, 21(2):551–565, 2013.

[81] Timothy Nelson, Daniel J. Dougherty, Christopher Barratt, and Kathi Fisler. The Margrave Tool for Firewall Analysis. In *LISA10 : 24th USENIX Conference on Large Installation System Administration*, San Jose, CA, USA, November 2010. USENIX.

[82] Haohui Mai, Ahmed Khurshid, Rachit Agarwal, Matthew Caesar, P. Brighten Godfrey, and Samuel Talmadge King. Debugging the data plane with anteater. *ACM SIGCOMM Computer Communication Review*, 41(4):290–301, October 2011.

[83] Ehab Al-Shaer and Mohammed Noraden Alsaleh. ConfigChecker: A tool for comprehensive security configuration analytics. In *Proc. of the 4th Symposium on Configuration Analytics and Automation (AFECONFIG2011)*, pages 1–2, Arlington, VA, October 2011. IEEE.

[84] Miroslav Sveda, Ondrej Rysavy, and Gayan De Silva. Static Analysis of Routing and Firewall Policy Configurations. In *Proc. of the 7th International Joint Conference (ICETE10)*, pages 39–53, Athens, Greece, October 2012. Springer Nature.

[85] Peyman Kazemian, George Varghese, and Nick McKeown. Header space analysis: Static checking for networks. In *Proc. of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI12)*, pages 9–9, San Jose, CA, April 2012. USENIX.

[86] Joshua D. Guttman. Filtering postures: Local enforcement for global policies. In *Proc. of the IEEE Symposium on Security and Privacy 1997*, pages 120 – 129, Oakland, CA, May 1997. IEEE.

[87] Joshua D. Guttman and Amy L. Herzog. Rigorous automated network security management. *International Journal of Information Security*, 4(1-2):29–48, February 2005.

[88] Alex Liu and Mohamed Gouda. Diverse Firewall Design. In *Int. Conference on Dependable Systems and Networks*, pages 595–604, Florence, Italy, June 2004.

[89] Alex Liu and Mohamed Gouda. Diverse Firewall Design. *IEEE Transactions on Parallel and Distributed Systems*, 19(9):1237–1251, September 2008.

[90] Alex Liu. Change-impact analysis of firewall policies. In *12th European Symposium On Research In Computer Security*, pages 155–170, Dresden, Germany, September 2007.

[91] Alex Liu. Firewall policy change-impact analysis. *ACM Transactions on Internet Technology*, 11(4):1–24, March 2012.

[92] Yi Yin and RS Bhuvaneswaran. Inferring the Impact of Firewall Policy Changes by Analyzing Spatial Relations between Packet Filters. In *Proc. of the Int. Conference on Communication Technology (ICCT06)*, pages 1–6, Guilin, China, November 2006. IEEE.

[93] Tomás E. Uribe and Steven Cheung. Automatic analysis of firewall and network intrusion detection system configurations. *Journal of Computer Security*, 15(6):691–715, July 2007.

[94] Nihel Ben Youssef, Adel Bouhoula, and Florent Jacquemard. Automatic verification of conformance of firewall configurations to security policies. In *Proc. of the 2009 IEEE Symposium on Computers and Communications*, pages 526–531, Sousse, Tunisia, July 2009. IEEE.

[95] Nihel Ben Youssef and Adel Bouhoula. Dealing with Stateful Firewall Checking. In *Communications in Computer and Information Science*, pages 493–507, Dijon, France, June 2011. Springer Nature.

[96] Jamie Jason, Lee Rafalow, and Eric Vyncke. IPsec Configuration Policy Information Model. RFC 3585 (Proposed Standard), August 2003. http://www.rfc-editor.org/rfc/rfc3585.txt.

[97] NSA's Information Assurance Directorate . NSA Mitigation Guidance. Technical report, 2016. https://www.nsa.gov/ia/mitigation_guidance/index.shtmls.

[98] National Institute of Standard and Technology . Recommendations of the national institute of standards and technology. Technical report, 2016. http://csrc.nist.gov/publications/PubsTC.html.

[99] International Organization for Standardization and International Electrotechnical Commission . ISO/IEC 27033: Information technology — security techniques — network security. Technical report, 2009. http://hsevi.ir/RI_Standard/File/11158.

[100] Cataldo Basile, Daniele Canavese, Christian Pitscheider, Antonio Lioy, and Fulvio Valenza. Assessing network authorization policies via reachability analysis. *Computers & Electrical Engineering*, pages –, 2017.

[101] Robert Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1972.

[102] Yosef Cohen and Jeremiah Y. Cohen. *Analysis of Variance*, pages 463–509. John Wiley & Sons, Ltd, 2008.

[103] Joel Halpern and Carlos Pignataro. Service Function Chaining (SFC) Architecture. RFC 7665, RFC Editor, October 2015. http://www.rfc-editor.org/rfc/rfc7665.txt.

[104] Scott Kelly and Sankar Ramamoorthi. Requirements for IPsec Remote Access Scenarios. RFC 3457, RFC Editor, January 2003. http://www.rfc-editor.org/rfc/rfc3457.txt.

[105] Paul Quinn and Tom Nadeau. Problem Statement for Service Function Chaining. RFC 7498, RFC Editor, April 2015. http://www.rfc-editor.org/rfc/rfc7498.txt.

[106] Adrian Farrel, Linda Dunbar, and Kathleen Moriarty. Interface to Network Security Functions (I2NSF). Technical report, Internet Engineering Task Force, 2015.

[107] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[108] Cataldo Basile, Antonio Lioy, and Stefano Paraboschi. *The PoSecCo Security Decision Support System*, pages 64–74. Springer Fachmedien Wiesbaden, Wiesbaden, 2012.

Appendix

Appendix A

Network Security Policy: Models for Policy Analysis

In this part of the thesis, we provide a detailed presentation of the mathematical models exploited in existing works of Policy Analysis, which we have mentioned in Chapter 3.

Decision Diagrams
A Binary Decision Diagram (BDD) is a digital function in terms of a directed, acyclic graph, which tells the user how to determine the output value of the function by examining the values of the inputs. The graph is composed of several decision nodes and terminal nodes. Each decision node is labelled by a Boolean variable, while the terminal nodes represent value 1 or 0. A BDD is called 'ordered' (OBBD) if different variables appear in the same order on all paths from the root. The BDD/OBDD is intended to provide a convenient means of finding the output of one or more functions for any given input.

A Interval Decision Diagram (IDD) is a generalization of BDD and MDD (Multi Decision Diagram), allowing diagram variables to be integers and child nodes to be associated with intervals rather than single values. Equivalent to BDDs, IDDs have a ordered form (OIDDs), providing a canonical representation of a functions which is important for formal verification.

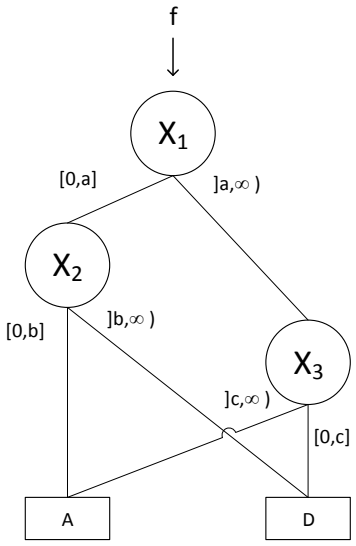


Fig. A.1 Example of an Interval Decision Diagram.

First-order logic

First-order logic (FOL) is a richer language than propositional logic, it contains not only the symbols $\wedge$, $\vee$, $\neg$, and $\rightarrow$ from propositional logic, but also the symbols $\exists$ and $\forall$ along with different symbols to represent variables, constants, and relations.

Logical consequence in FOL is only semi-decidable: if a sentence A implies a sentence B then this can be discovered, but if A does not imply B, this does not mean that A implies the negation of B.

A deductive system is used to demonstrate that one formula is a logical consequence of another formula. There are many such systems for FOL these share the common property that a deduction is a finite syntactic object; the format of this object, and the way it is constructed, vary widely.

Association Rule Mining

The Association Rule Mining was first introduced by Agrawal *et al.* [107]. An association rule is an implication of the form: $X \rightarrow Y$, where X, Y are two disjoint sets. The goal of Association Rule Mining is to extract correlations, frequent patterns, associations and/or casual structures among sets of items in the transaction databases.

Propositional Satisfiability Problem
The propositional satisfiability problem abbreviated as SATISFIABILITY (often called SAT) is the problem of determining whether a set of sentences in propositional logic is satisfiable.

In practice, many automated reasoning problems in propositional logic are first reduced to satisfiability problems and then solved by using a satisfiability solver. Today, SAT solvers are commonly used in hardware design, software analysis, planning, mathematics, security analysis, and many other areas.

Finite State Machine
A Finite State Machine (FSM) describe models that contains finite number of states and produces outputs on state transitions after receiving inputs. FSM are widely used to model systems in diverse areas.

A FSM m is a quintuple: $m = (I, O, S, \delta, \lambda)$. Where:

- I is the finite set of symbols representing input to m;
- O is the finite set of symbols representing output to m;
- S is the finite set of symbols representing states of m;
- $\delta : S \times I \to S$ is the state transition function ;
- $\lambda : S \times I \to O$ is the output function.

When the machine is in a current state s_1 in S and receives an input i_1 a from I it moves to the next state $s_2 = \delta(s_1, i_1)$ and produces an output $o_1 = \lambda(s_1, i_1)$.

An FSM can be represented by a State Transition Diagram (STD). An STD is a directed graph where the vertices are the states of the machine and the edges are the state transitions. In addition each edge is labelled with the input and output associated with the transitions.

Geometrical model
The geometrical model represent a policy is a function represented as a four-tuple $(R, \mathfrak{R}, E, a_d)$, where:

- $R = \{r_i\}_i$, $i \in [1, n]$ is the rule set.
- $\mathfrak{R} : 2^R \rightarrow \mathcal{A}$ is the resolution function used to decide the action for packets matching more than one rule. An example of resolution function is the First Matching Rule strategy (FMR) that, in case of multiple matching rules, selects the action from the rule at highest priority. The firewall action set includes "allow" and "deny" actions and will be indicated as $\mathcal{A} = \{a, d\}$.
- $E = \{E_1, E_2, \ldots\}$ is the set of external data associated to the rules. External data are not part of the rules, nevertheless they are used to take decisions. The association is done using a set of external data functions $\varepsilon_k : R \rightarrow E_k$. FMR uses priorities as external data. The function that associates rules to priorities will be denoted by π.
- a_d is the default action, applied when a packet matches no rules.

Rules $r_i = (c_i, a_i)$ are composed of a condition clause c_i and an action $a_i \in \mathcal{A}$. The conditions clause is defined as:

$$c_i = s_{i1} \times \cdots \times s_{i1} \subseteq S_{i1} \times \cdots \times S_{im} = \mathcal{S}$$

where each condition s_{ij} is a subset of a *selector* S_{ij}. Examples of selectors are the protocol fields mentioned previously, like IP source address and port number. The set $\mathcal{S} = S_{i1} \times \cdots \times S_{im}$ is named *decision space*. The condition clause c_i is thus a hyper-rectangle (or the union of hyper-rectangles) in $\mathcal{S}$.

Appendix B

Communication Protection Policies: Validation Materials

We now provide additional materials regarding the validation of our analysis model of Communication Protection Policies, presented in Chapters 7 and. In particular, we present the configurations of some selected security controls and the mapping of their configuration into our model as a set of PIs (see Chapter 5) to prove the expressiveness of our approach and the easiness of the translation.

B.1 Example of system configuration

Translating data and channel protection configurations into/from the policy representations we use in our model is a straightforward task. The policy implementations convey in a very compact way all the information that is needed to correctly configure the security controls. To prove our claim, we provide the mapping of configurations of three different technologies, using the scenario depicted in Fig. B.1[1]: 1) IPsec, by mapping a strongSwan[2] configuration for an end-to-end, site-to-site and remote access channels; 2) TLS, by mapping an OpenVPN[3] configuration to establish a secure tunnel; 3) SSH, by mapping a SSH tunnel configuration. Analogously, all the configurations of the

[1]The figure was retrieved from https://www.strongswan.org/test-scenarios.html.
[2]See https://www.strongswan.org/.
[3]See https://openvpn.net/index.php/open-source/documentation/.

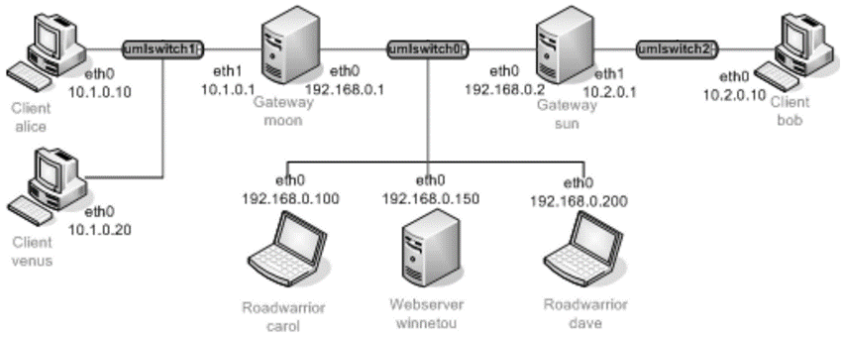


Fig. B.1 Network scenarios.

secure communications used in our analysis model can be mapped into policy implementations.

B.1.1 IPsec configurations (strongSwan)

Listing B.1 shows a strongSwan end-to-end connection from the host 192.168.0.100 to the IP address 192.168.0.200, which requires aes256-sha512-modp2048 as ESP parameters.

```
config setup
 conn %default
 ikelifetime=60m
 keylife=20m
 rekeymargin=3m
 keyingtries=1
 keyexchange=ikev1
conn end2end
 left=192.168.0.100
 leftcert=moonCert.pem
 leftid=@moon.strongswan.org
 leftfirewall=yes
 right=192.168.0.200
 rightid=@sun.strongswan.org
 ike=aes256-sha512-modp2048
 esp=aes256-sha512-modp2048
```

Listing B.1 End-to-end strongSwan configuration.

This configuration is summarized by the following PI:

$$(192.168.0.100, 192.168.0.200, \text{IPsec}, (5, 5, 5), *, \varnothing)$$

Listing B.2 presents a site-to-site configuration between the subnets 10.1.0.0/16 and 10.2.0.0/16 operated by the gateways whose IP addresses are 192.168.0.1 and 192.168.0.2, respectively.

```
conn %default
 ikelifetime=60m
 keylife=20m
 rekeymargin=3m
 keyingtries=1
 keyexchange=ikev1

conn site2site
 left=192.168.0.1
 leftcert=moonCert.pem
 leftid=@moon.strongswan.org
 leftsubnet=10.1.0.0/16
 leftfirewall=yes
 right=192.168.0.2
 rightid=@sun.strongswan.org
 rightsubnet=10.2.0.0/16
 ike=aes256-sha512-modp2048
 esp=aes256-sha512-modp2048
```

Listing B.2 Site-to-site strongSwan configuration.

This configuration can be represented with the following PI:

$$(192.168.0.1, 192.168.0.2, \text{IPsec}, (5, 5, 5), (10.1.0.0/16, *, 10.2.0.0/16, *, *), \varnothing)$$

Finally, Listing B.3 configures a remote access for the host 192.168.0.100 to allow access the subnet 10.2.0.0/16 through the gateway 192.168.0.1.

```
conn %default
 ikelifetime=60m
 keylife=20m
 rekeymargin=3m
 keyingtries=1
 keyexchange=ikev1

conn remoteAccess
 left=192.168.0.100
 leftsourceip=%config
 leftcert=carolCert.pem
 leftid=carol@strongswan.org
 leftfirewall=yes
 right=192.168.0.1
 rightsubnet=10.1.0.0/16
 rightid=@moon.strongswan.org
 ike=aes256-sha512-modp2048
 esp=aes256-sha512-modp2048
```

Listing B.3 Remote access strongSwan configuration.

The policy implementation for this configuration is:

$$(192.168.0.100, 192.168.0.1, \text{IPsec}, (5, 5, 5), (*, *, 10.2.0.0/16, *, *), \varnothing)$$

B.1.2 TLS (OpenVPN)

Listing B.4 shows a configuration for OpenVPN of a tunnel from the generic client 192.168.0.100 to the server 192.168.0.150:1194 (Webserver), which requires the enforcement of a cipher-suite that uses Diffie-Hellman with AES-256-CBC and SHA512.

```
client
dev tun
proto udp
remote Webserver 192.168.0.150:1194
nobind
ca ca.crt
cert client.crt
key client.key
remote-cert-tls server
tls-auth ta.key 1
cipher AES-256-CBC
auth SHA-512
dh dh1024.pem
```

Listing B.4 Client side OpenVPN 2.0 configuration.

This configuration translates into the following PI:

$$(192.168.1.100 : *, 192.168.0.150{:}1194, \text{TLS}, (5, 5, 5), *, \varnothing)$$

Listing B.5 shows the relative OpenVPN configuration of the server.

```
local 192.168.0.150
port 1194
dev tun
proto udp
keepalive 10 120
ca ca.crt
cert server.crt
key server.key
tls-auth ta.key 0
cipher AES-256-CBC
auth SHA-512
dh dh1024.pem
```

Listing B.5 Server side OpenVPN 2.0 configuration.

B.1.3 SSH

Listing B.6 presents the configuration of an SSH tunnel for the user whose username is 'client' and its IP address is 192.168.0.200. The user connects to the SSH server at 192.168.0.150:22022, which uses AES256-CBC and SHA512 to secure the data and enters the local network 10.0.0.0/16 with the 10.0.0.3 IP address on port 3306.

```
Host tunnel
 #SSH connection setting
 HostName 192.168.0.150
 User dave
 Port 22022
 IdentityFile ~/.ssh/client.example.key
 Ciphers aes256-cbc
 MACs hmac-sha2-512

 #SSH tunnel setting
 LocalForward 10.0.0.3:3306 127.0.0.0:3306
```

Listing B.6 Client side SSH configuration.

The PI corresponding to this SSH configuration is:

$$(192.168.2.100 : *, 192.168.2.1{:}22022, \text{SSH}, (5,5,5),$$
$$(10.0.0.3, 8080, 192.168.2.1, 3306, TCP), \varnothing)$$

B.2 Graphical user interface

A graphical user interface was developed referring to the model representation described in Chapter 6. In the interface, PI are called Logica Association Implementation (LAimpl)[4].

[4]The GUI was developed in the context of the European protect PoSeCo[108].

Fig. B.2 Graphical user interface.

Fig. B.3 Graphical user interface.